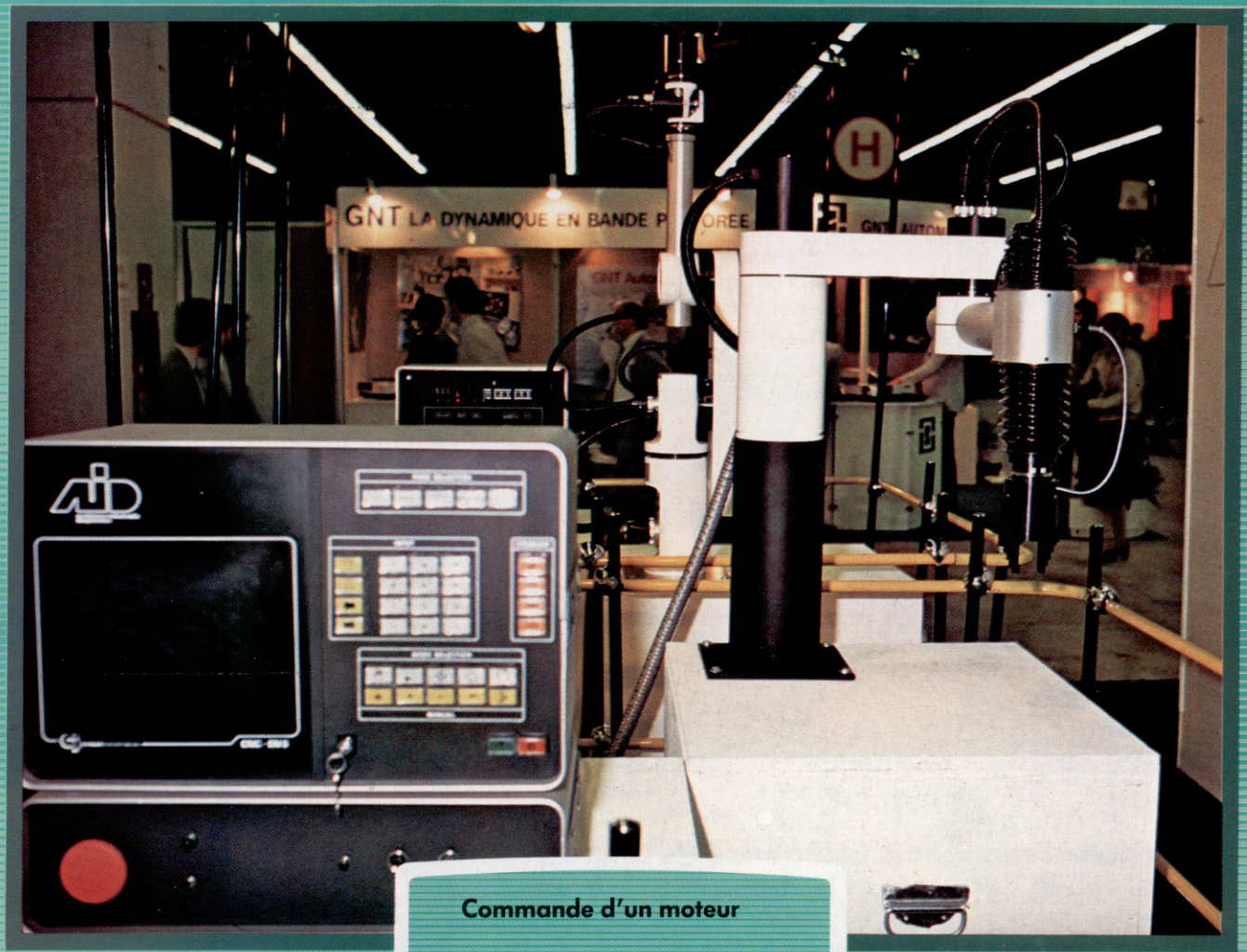
ABC INFORMATIQUE

N°54

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE



Commande d'un moteur

Manches à balai à infrarouges

Logo (suite)

Accélérer le BASIC

EDITIONS ATLAS

M 6062-54-12-F 85 FB - 3,80 FS - $ 1.95

Page manquante
(publicité et colophon)

Finale

Dans cette dernière partie de la série, nous examinons certains systèmes perfectionnés, qui ont fait leur apparition depuis que la musique électronique a commencé à intégrer la technologie numérique.

Le développement récent le plus significatif ayant marqué la musique électronique touche au domaine de l'enregistrement numérique. Non seulement la qualité d'enregistrement du son s'est améliorée de manière considérable, mais encore le terme « enregistrement » a changé de signification : il concerne maintenant de nombreuses et différentes techniques. Si nous comprenons que le terme enregistrement signifie aujourd'hui « codage numérique du son et son organisation en musique », nous commençons à saisir ce qui s'est produit en musique dans les années quatre-vingt.

Depuis la Seconde Guerre mondiale, on a établi une norme pour l'enregistrement sur bande magnétique, et les formats vont de la toute petite microcassette aux grandes bobines de vingt-quatre pistes utilisées dans les studios d'enregistrement professionnels.

Lors d'un enregistrement, les minuscules particules d'oxyde métallique sont redisposées dans des configurations complexes, qui sont analogues aux formes des ondes sonores qu'elles représentent. Lorsque la bande défile devant la tête de lecture d'une enregistreuse, ces configurations sont converties en une chaîne de tensions électriques. Ces tensions sont alors transmises aux haut-parleurs, qui reproduisent le son enregistré sur la bande.

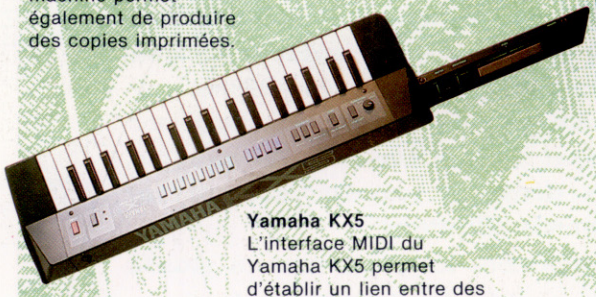
Puisque la disposition des particules peut être assez précisément appréciée par rapport à la tête de lecture, il est assez facile de savoir où se trouve un son particulier sur une section de bande. L'édition de la bande à l'aide de lames de rasoir démagnétisées est donc devenue une importante technique qui doit être maîtrisée par les ingénieurs du son.

En enregistrement numérique, le son est codé numériquement sur la bande, et la tête de lecture devient un convertisseur numérique/analogique (N/A). Les haut-parleurs sont alimentés de la même manière qu'auparavant, mais ils utilisent les tensions générées par le convertisseur N/A. Pourvu que les données de conversion soient assez nombreuses, la bande numérique peut reproduire des sons d'une qualité bien supérieure à celle obtenue avec une bande magnétique. Et, aussi longtemps que les données demeurent intactes, la bande peut être copiée numériquement des centaines de fois, sans perte de qualité.

Ian McKinnell

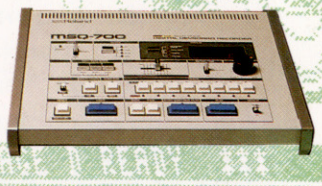
Synclavier
Le Synclavier de New England Digital est considéré comme étant l'une des unités les plus perfectionnées au monde. En plus des fonctions habituelles du synthétiseur, la machine a la possibilité de stocker jusqu'à 10 mégaoctets de son sur disque dur.

Fairlight CMI
Le Fairlight CMI fut l'un des premiers systèmes de musique utilisant un ordinateur. Le système d'exploitation du Fairlight est piloté par menus, ce qui offre une variété d'options allant de la commande au clavier jusqu'à la définition des formes d'ondes. Cette machine permet également de produire des copies imprimées.



Yamaha KX5
L'interface MIDI du Yamaha KX5 permet d'établir un lien entre des synthétiseurs et le Yamaha CX5.
Ce dispositif servira aussi à établir l'interface de l'ordinateur domestique Yamaha MSX.

Roland MSQ-700
Le Roland MSQ-700 est considéré comme le meilleur séquenceur compatible avec MIDI. Le MSQ-700 offre la gamme complète des fonctions MIDI et peut stocker jusqu'à 6 500 notes.



Drumulator
La batterie électronique Drumulator d'Emu Systems a une capacité mémoire de 10 088 notes à l'intérieur de 64 chansons. Il est aussi possible d'insérer des ROM additionnelles pour produire des effets spéciaux, comme des percussions latines ou africaines.

De nombreux studios d'enregistrement se sont dotés d'enregistreuses numériques à vingt-quatre pistes. A l'aide de celles-là, la reproduction du son est si précise que même les ingénieurs du son ont du mal à déterminer si un son provenant des haut-parleurs du studio est produit par un instrument de musique, dans la zone d'enregistrement, ou par une bande numérique. Mais de nouveaux problèmes sont apparus : il n'est plus possible de « voir » où se trouvent les sons sur la bande numérique, ce qui rend l'édition beaucoup plus difficile. Le collage devient une technique dépassée. Le « bruit de studio » représente une autre difficulté : il s'agit d'une caractéristique indésirable, et généralement inaudible, de certains équipements de studio. La bande magnétique n'était pas assez sensible pour l'enregistrer, mais les enregistrements numériques ont tendance à la capter.

Tandis que l'enregistrement sur vingt-quatre pistes est toujours imposé par les studios professionnels, l'enregistrement numérique monopiste d'une qualité identique est offert à tout propriétaire de magnétoscope domestique Betamax. La bande vidéo est un médium numérique, elle peut donc être utilisée pour coder tout type de données. Le Sony PCM (Pulse Code Modulator) est une unité qui transforme un magnétoscope Betamax en une enregistreuse de bande audio. Cette unité peut rendre désuète les enregistrements analogiques de même taille.

Le codage numérique du son, ou échantillonnage, est le principe essentiel du Fairlight CMI (Computer Musical Instrument), l'un des systèmes perfectionnés les plus connus. Le Fairlight peut échantillonner tout son pendant une période allant jusqu'à deux secondes, et il reproduit alors ce son sur une plage de six octaves. L'échantillonnage représente une véritable percée dans le domaine de la musique électronique. Pendant des années, les ingénieurs et les musiciens ont tenté de simuler les sons des instruments à cordes ou à vent à l'aide de synthétiseurs, et, dans certains cas, ils ont été très près d'atteindre leur but. Mais l'échantillonnage ne permet pas uniquement une remarquable reproduction d'un son d'instrument à cordes. Il peut produire le son d'un violon particulier. Il peut même reproduire le son d'un joueur particulier dans une salle particulière. Dans un précédent article, nous avons vu comment les compositeurs de musique concrète des années cinquante passaient des semaines à assembler des fragments de bande magnétique pour produire des œuvres complètes. La manipulation des échantillons par ordinateur permet maintenant à un compositeur d'obtenir le même résultat en quelques minutes.

Échos du passé
La description des sons faite par Francis Bacon (1561-1626) dans cet extrait semble prédire l'extraordinaire force et variété de la musique électronique d'aujourd'hui. (Cl. Mary Evans.)

D'après "The New Atlantis" de Francis Bacon, publié en 1624.

Nous avons aussi les maisons du bruit, où nous produisons, décrivons et expliquons tous les sons. Nous avons des harmonies que vous n'avez pas, en quart de ton et en petits coulés musicaux. Divers instruments de musique également que vous ne connaissez pas, certains plus doux que n'importe lequel de ceux que vous possédez, qui associés aux cloches et sonneries donnent des sons d'une grande délicatesse.

Nous transformons les petits sons en grands bruits graves et profonds, et les grands bruits en petits sons atténués et aigus. Nous produisons divers tremblements de sons qui, mêlés à des gazouillis, semblent tout à fait naturels. Nous imitons et répétons tous les sons articulés, toutes les voix et tous les cris d'animaux et d'oiseaux, et nous utilisons des systèmes qui facilitent l'écoute. Nous avons aussi divers, étranges et artificiels échos qui réfléchissent la voix plusieurs fois, comme si elle était ballotée dans l'espace et d'autres, encore, déchirent la voix, restituant les sons articulés dans un ordre différent de celui reçu. Nous avons aussi les moyens de transmettre le son, par le biais de tubes et de tuyaux, à distance et selon diverses lignes musicales.

Instruments d'échantillonnage

Un instrument d'échantillonnage comme le Fairlight peut remédier aux limites naturelles des instruments musicaux. Par exemple, il est assez facile pour un flûtiste de produire, avec un souffle de qualité, un son chaud, dans le registre inférieur de la flûte. Il lui sera cependant impossible d'obtenir ce type de son deux octaves plus haut, au registre supérieur de l'instrument — la conception physique de la flûte l'interdisant. En revanche, un utilisateur de Fairlight peut échantillonner le son grave de qualité et le transposer deux octaves plus haut sur le clavier. Le résultat ressemblera toujours au son d'une flûte, mais dont l'équivalent réel ne peut exister.

Le Fairlight peut produire un affichage sur écran de tous les sons échantillonnés, qui sont stockés sur une disquette de 8 pouces. Les différentes caractéristiques d'un son individuel peuvent être examinées successivement : il est souvent plus facile de déterminer ce qui est « mauvais » dans un son en l'examinant de cette façon, plutôt qu'en l'écoutant. De nombreux sons doivent être plus longs que la durée d'échantillon (de deux secondes). En constatant comment sont associées les différentes formes d'ondes du son, il est possible de sélectionner un point où le son pourrait être répété. Si on sélectionne le point approprié, une illusion de continuité peut être produite. Le son analogique ne se comporte évidemment pas comme cela. Aussi, la répétition du son choisi pourrait donner une dimension inusitée à la musique ainsi produite.

L'utilisateur du Fairlight peut entrer la musique de deux manières, en plus du jeu « en temps réel » sur le clavier. La première méthode, dite

« Page R », affiche une portée de cinq lignes, et l'utilisateur entre les notes sur la portée à partir du clavier musical. Toute erreur de synchronisation peut être réparée automatiquement par l'ordinateur en se basant sur la mesure déjà définie par l'utilisateur.

La seconde méthode implique l'utilisation d'un langage de composition musical (MCL). Le MCL du Fairlight nécessite l'entrée de chaque note au clavier alphanumérique, mais il permet de modifier les différents paramètres au niveau de chaque note. Le Fairlight a une capacité de huit voix, un utilisateur peut donc entrer huit séquences différentes, jouées par huit « instruments » différents. Chacun de ceux-ci peut être très légèrement décalé par rapport aux autres, et l'exécution globale est coordonnée par l'horloge interne du Fairlight.

On peut se demander pourquoi une musique doit être exécutée de façon aussi « imprécise », surtout par un ordinateur. C'est que les musiciens ne jouent jamais en conservant exactement le même tempo. Le système Fairlight permet de simuler certains styles d'exécution.

De nombreux musiciens s'inquiètent du fait que le Fairlight sera de plus en plus en mesure de les remplacer, la capacité de simulation de tels équipements continuant à évoluer. Des groupes comme Wang Chung, Duran Duran et Culture Club utilisent des Fairlight dans leur processus de réalisation, et il est souvent impossible de distinguer ce qui est vraiment joué de ce qui est exécuté par le Fairlight.

Bien qu'étant le plus connu, le Fairlight n'est pas le seul instrument de ce type offert sur le marché. Le système Synclavier dispose de fonctions similaires, mais à une plus grande échelle. Les données sont stockées sur un disque dur d'une capacité de 40 méga-octets. Un disque complet peut être produit et enregistré à l'intérieur du système Synclavier, ce qui rend complètement inutile la présence d'une machine numérique à vingt-quatre pistes.

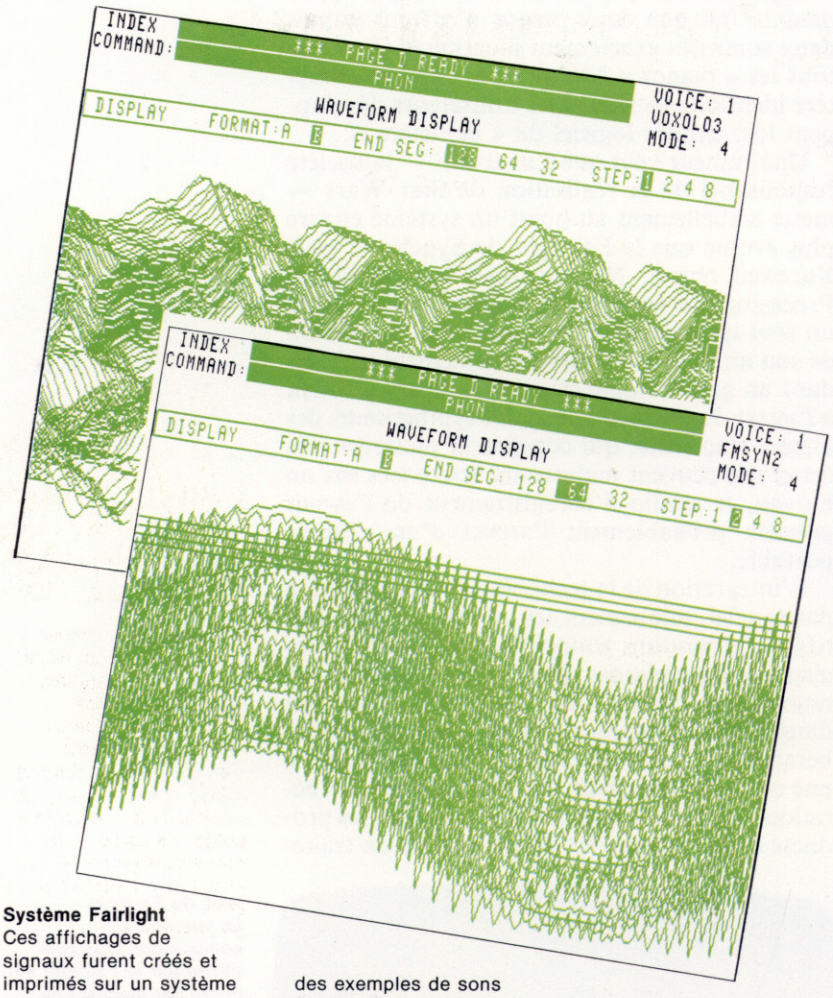
Mais même le système Synclavier souffre de certaines restrictions. Actuellement, tous les instruments d'échantillonnage disponibles ont une chose en commun : ils reproduisent le son échantillonné de façon aussi précise que possible — sauf si l'utilisateur est intervenu pour faire une modification spécifique. Mais un trompettiste, en pleine représentation, peut apporter une modification importante au son suivant en changeant simplement l'attaque du souffle ou la position des lèvres. Un bon musicien y parvient presque sans y penser. Les instruments d'échantillonnage doivent donc également produire un système qui obéit rapidement à la volonté du musicien. Le Kurzweil, qui n'est encore qu'un prototype, comporte un programme de reconnaissance de configuration. Cela signifie que, lorsqu'une note est jouée sur le clavier musical, de nombreux échantillons différents sont analysés et que les caractéristiques de chaque échantillon sont combinées pour produire le son individuel. Un tel système permet de se rapprocher beaucoup plus du caractère et de la nature d'un véritable instrument de musique. La seule différence réside dans le fait que deux pianos n'offrent jamais deux sonorités exactement identiques alors que tous les « pianos » Kuzweil CMI ont un caractère identique — sauf si les utilisateurs développent leur propre logiciel de « caractère ».

Une rumeur veut que Lucasfilm — la société responsable de la réalisation de Star Wars — mette actuellement au point un système encore plus évolué que le Fairlight, le Synclavier et le Kurzweil réunis. Nommé ASP (Audio Signal Processor), cet équipement est censé intégrer dans un seul système toutes les fonctions de gestion de son musical numérique actuellement offertes dans un grand studio musical complexe. Ainsi, à l'instar des ordinateurs et des synthétiseurs des années cinquante, qui occupaient plusieurs pièces et qui peuvent maintenant être posés sur un bureau, le studio d'enregistrement de l'avenir prendra probablement l'aspect d'un système portable.

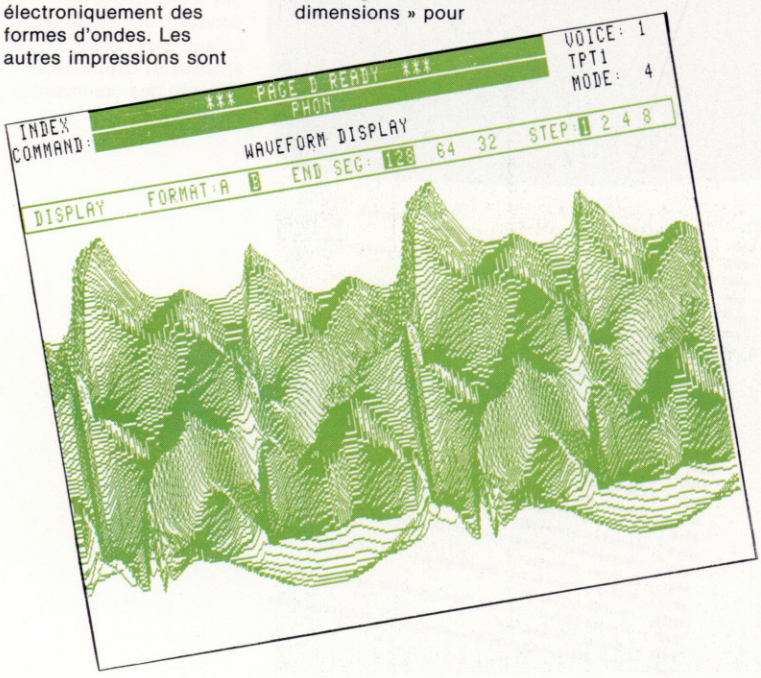
L'intégration de la technologie numérique n'a pas touché uniquement les systèmes et dispositifs de génération sonore. Les studios d'enregistrement modernes disposent en général de nombreuses unités de traitement du son dans leur équipement de base. L'unité de réverbération est un bon exemple. Le son traverse une unité qui lui ajoute un écho ou une réverbération, et les guitaristes de rock, ainsi que les producteurs de reggae ont utilisé ce type de traite-


```
0001  !B=24:O=1/24:V=15
0002  R,36:<(B♯4:D4),5:(B♯4:D4),7
0003  R,5:(B♯4:D4)V10,7/3:(B♯4:D4)V11,7/3:(F♯4:C4)V13,7/3
0004  (B♯4:D4),5:(B♯4:D4),7
0005  R,5:(B♯4:D4)V10,7/3:(B♯4:D4)V11,7/3:(F♯4:C4:E♯4)V13,7/3
0006  (D4:F5),5:(D4:F5),7
0007  R,5:(D4:F5)V10,7/3:(D4:F5)V11,7/3:(G4:B4:E♯4)V13,7/3
0008  (B♯4:D4),5:(B♯4:D4),7
0009  R,5:(B♯4:D4)V10,7/3:(B♯4:D4)V11,7/3:(F♯4:C4)V13,7/3>2
0010  <(B♯4:D4),7/3:(B♯4:D4),7/3:(F♯4:C4),7/3:T=+3>3:T=0
0011  <(B♯4:D4:A5),5:(C4:E♯4),7
0012  (B♯4:D4:A5),5:(C4:E♯4),7/2:(D4:B♯5),7/2>2
0013  (C4:E♯4:B♯5),5:(B♯4:D4),7
0014  (C4:E♯4:B♯5),5:(B♯4:D4),7/3:(E♯4:C4),7/3:(A5:C5),7/3
0015  (C4:E♯4:B♯5),5:(B♯4:D4),7
0016  (C4:E♯4:B♯5),5:(B♯4:D4),7/4:R,5/6
0017  (B♯4:D4),7/8:R,5/12
0018  (G4:B4:E♯4),75/24:V=15
0019  <(B♯4:D4),5:(B♯4:D4),7
0020  R,5:(B♯4:D4)V10,7/3:(B♯4:D4)V11,7/3:(F♯4:C4)V13,7/3
0021  (B♯4:D4),5(B♯4:D4),7
0022  R,5:(B♯4:D4)V10,7/3:(B♯4:D4)V11,7/3:(F♯4:C4:E♯4)V13,7/3
0023  (D4:F5),5:(D4:F5),7
0024  R,5:(D4:F5)V10,7/3:(D4:F5)V11,7/3:(G4:B4:E♯4)V13,7/3
0025  (B♯4:D4),5:(B♯4:D4),7
0026  R,5:(B♯4:D4)V10,7/3:(B♯4:D4)V11,7/3:(F♯4:C4)V13,7/3>2
```

MCL du Fairlight
Le langage de composition musicale utilisé par le Fairlight est piloté par menus. Les choix sont ainsi effectués à partir de menus proposés à l'écran. Les signaux affichés à la page 1064 furent générés à l'aide d'une liste de paramètres sonores dans des instructions de données, similaires à un listage BASIC, puis affichés et imprimés en utilisant des commandes du menu.



Système Fairlight
Ces affichages de signaux furent créés et imprimés sur un système musical Fairlight CMI. Le premier est un exemple d'ondes sinusoïdales générées par le Fairlight à l'aide de la synthèse FM. Ce son fut fabriqué en combinant électroniquement des formes d'ondes. Les autres impressions sont des exemples de sons échantillonnés. Ces signaux sont produits en numérisant les sons réels d'une voix humaine, pour la deuxième illustration, et d'une trompette pour la troisième. Les affichages sont en « trois dimensions » pour illustrer les modifications apportées à la composition de chaque son à travers le temps.



ment pour obtenir leur son musical particulier. Le quantec est une unité numérique qui, au lieu de produire uniquement une réverbération, réussit à simuler les types de réverbération qui se manifestent dans des pièces et dans des espaces de dimensions différentes. Son « espace acoustique » le plus petit est une boîte d'un volume de 1 m^3, et les simulations prédéfinies offrent les dimensions d'un salon, d'une salle de concert, d'un hangar d'aviation ou bien d'une cathédrale.

La fonction la plus intéressante du quantec est d'étendre la réverbération bien au-delà des limites physiques et acoustiques naturelles. Dans le Midwest américain, au cours des années cinquante, un petit studio de rockabilly avait été construit près d'un silo à céréales. C'est cet énorme espace qui permettait de créer le son particulier du rockabilly dans ce studio. L'utilisation moderne d'unités numériques comme le quantec est plus banale, étant réservée principalement aux opérations de postproduction et au travail de cinéma.

Les acteurs peuvent être filmés tout en parlant dans l'environnement acoustiquement neutre d'un studio, et la bande sonore sera traitée ultérieurement afin d'obtenir une résonance acoustique qui corresponde à l'espace dans lequel l'action est censée se dérouler.

Le défi de la musique électronique

Nombreux sont ceux qui ont le sentiment que la technologie numérique a une mauvaise influence sur la musique. Ils pensent que cette musique est créée d'une manière tellement artificielle que la spontanéité et la créativité sont délaissées au profit d'un meilleur contrôle technique. C'est un argument persuasif. Mais prenons la peine d'examiner un médium visuel — le cinéma — pour juger si cela est vraiment fondé.

Depuis plusieurs décennies, la technologie cinématographique a permis aux metteurs en scène de filmer des scènes sous différents angles, d'utiliser des zooms puissants et d'effectuer de larges panoramiques. Il a été possible de répéter des séquences d'images, de les ralentir, de les accélérer, de les faire défiler en sens inverse, au cours du montage du film, pour créer des juxtapositions d'images les plus diverses. Ce n'est que tout récemment, grâce à la technologie numérique, que les producteurs de musique ont commencé à vraiment contrôler le son. Le cinéma est généralement reconnu comme un médium permettant une libre expression. Il est donc problable que la musique numérique sera considérée de la même manière. Actuellement, certains musiciens sont facilement un peu intimidés par tant de possibilités, et d'autres produisent une musique qui n'offre guère d'autre intérêt que technique. Pour les musiciens qui sont prêts à relever le défi de la musique électronique, les années quatre-vingt s'annoncent comme une décennie stimulante et créative.

Commande d'un moteur

Nous découvrons maintenant le logiciel requis pour mettre les moteurs sous tension, et nous développons un système simple de commande à rétroaction.

La commande de dispositifs par microprocesseurs est de nos jours fort répandue dans l'industrie. Ses applications vont du comptage de bouteilles sur un tapis roulant à la soudure des châssis d'automobiles. Tous les systèmes de commande doivent d'abord assurer l'entrée de données provenant de l'extérieur sous une forme reconnue par un système à microprocesseur, ensuite analyser ces données et, enfin, provoquer les actions requises par cette analyse. Si ces trois interventions sont répétées de façon continue, le système assure une commande dite « à rétroaction ».

Pour illustrer le principe de la commande à rétroaction, prenons l'exemple d'une casserole de soupe mijotant sur une cuisinière. Afin de cuire la soupe, la cuisinière doit fournir assez de chaleur pour la faire légèrement bouillir, sans qu'elle déborde de la casserole. Pour ce faire, nous n'aurions qu'à appliquer une température maximale à la casserole jusqu'à ce que la soupe commence à bouillir; alors, il suffirait de baisser le feu de la plaque pour que la soupe mijote. Si elle recommençait à bouillir, nous devrions diminuer encore la chaleur de la plaque de la cuisinière. En effectuant ce type d'opérations, nous contrôlons visuellement l'état de la soupe, nous analysons ce que nous voyons, et nous prenons les mesures appropriées. Nous devons répéter ces interventions jusqu'à ce que la soupe soit cuite. Un micro-ordinateur pourrait contrôler la cuisson, mais d'une autre façon. La principale différence résiderait dans le mode de contrôle de l'état de la soupe. Alors que nous pouvons regarder la soupe et prendre une décision en nous basant sur notre expérience, un système informatique devrait utiliser une autre méthode, basée sur des données physiques facilement vérifiables, comme la température. Avant d'automatiser le contrôle de la soupe, il est nécessaire d'effectuer certaines expériences pour déterminer quelle température correspond à un mijotement régulier, et à quelle température la soupe déborderait de la casserole. Cela étant fait, l'ordinateur pourrait se charger complètement de cette tâche, pourvu que l'on ait des dispositifs adéquats pour contrôler et pour régulariser la température.

Des moteurs peuvent être commandés de diverses manières en utilisant le boîtier tampon et le boîtier de sortie à basse tension que nous avons construits. Un moteur de modèle réduit est idéal pour tester le logiciel que nous allons concevoir. Nous devons d'abord nous assurer que le moteur que nous connectons au boîtier de sortie a une valeur nominale de tension égale ou supérieure à celle de la tension d'entrée provenant du transformateur. En connectant les deux bornes du moteur à la ligne 0 du boîtier de sortie, nous pouvons mettre le moteur sous tension et hors tension en utilisant les touches Z et X du clavier.

BBC MICRO

```
10 REM SIMPLE MOTEUR BBC
20 DDR=&FE62:DATREG=&FE60
30 ?DDR=255:REM TOUTES LES LIGNES DE SORTIE
40 ?DATREG=0:REM MOTEUR HORS TENSION
50 REPEAT
60 A$=INKEY$(1):REM FRAPPE D'UNE TOUCHE?
70 IF A$="Z" THEN ?DATREG=1:REM MISE SOUS TENSION
80 UNTIL A$="X"
90 ?DATREG=0:REM MISE HORS TENSION
```

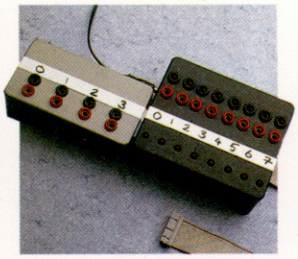
COMMODORE 64

```
10 REM SIMPLE MOTEUR CBM 64
20 DDR=56579:DATREG=56577
30 POKEDDR,255:REM TOUTES LIGNES DE SORTIE
```

Exercices

Nous pouvons maintenant mettre au point certaines expériences de commande qui impliquent une entrée et une sortie utilisant le boîtier tampon et le boîtier de sortie à basse tension. Les capteurs mentionnés dans les expériences suivantes peuvent être des capteurs à pression ou des interrupteurs à lamelle utilisant un petit aimant. Vous pouvez facilement vous procurer ces dispositifs auprès de fournisseurs en matériel électrique et électronique. Les interrupteurs thermosensibles fonctionnent en établissant un contact interne entre les bornes lorsqu'une certaine température est atteinte; ils peuvent être achetés de la même façon.

1) Écrivez un programme qui fait reculer et avancer un véhicule entre deux capteurs placés sur son parcours.
2) Écrivez un programme qui fait arrêter un véhicule directement au-dessus d'un capteur.
3) Écrivez un programme qui maintienne un récipient d'eau entre 70 °C et 100 °C en utilisant un chauffe-eau à basse tension et deux interrupteurs thermosensibles.
4) Écrivez un programme qui calcule la vitesse en mètres par seconde d'un véhicule allant d'un point à un autre. (Vous devrez connaître la distance séparant les deux points et enregistrer le temps requis pour parcourir cette distance.)



Mettre d'autres tampons Le boîtier tampon, notre première construction, sert à protéger les circuits de l'ordinateur contre des courants d'entrée ou de sortie trop élevés. Son alimentation indépendante alimente le boîtier de sortie — le projet plus récent — et autorise l'interruption commandée par logiciel de la sortie de 12 V. (Cl. Ian McKinnell.)

```
40 POKEDATREG,0:REM MOTEUR HORS TENSION
50 GETA$:IFA$<>«Z» ANDA$<>«X» THEN 50
   :REM ATTENDRE LA FRAPPE D'UNE TOUCHE
60 IF A$=«Z»THEN POKEDATREG,1:GOTO 50
70 IF A$=«X»THEN POKEDATREG,0:END
```

Nous pouvons commander la direction du moteur en connectant les bornes du moteur à des lignes adjacentes sur le boîtier de sortie. Le diagramme illustre ces connexions. Nous pouvons également utiliser un simple interrupteur à rupture, connecté à la ligne 7 du boîtier tampon pour commander la direction.

Dans le programme suivant, une valeur de 1 dans le registre de données entraîne dans le moteur une circulation de courant dans un sens. Placer une valeur de 2 dans le registre de données fait circuler le courant dans la direction inverse. Le programme teste de façon répétitive la ligne 7 et ne place un 2 dans le registre de données que lorsque la ligne est au niveau bas (c'est-à-dire lorsque l'interrupteur est fermé). De cette façon, l'interrupteur commande la direction du moteur. Voici un exemple très simple de système de commande à rétroaction.

BBC MICRO

```
10 REM MOTEURS DIRIGÉS BBC
20 DDR=&FE62:DATREG=&FE60
30 ?DDR=127:REM ENTRÉE LIGNE 7
40 ?DATREG=0:REM MISE HORS TENSION
50 A$=GET$:REM ATTENDRE LA FRAPPE D'UNE TOUCHE
60 REPEAT
70 IF A$=INKEY$(1)
80 IF(?DATREG AND128)=0 THEN DIR=2 ELSE DIR=1
90 ?DATREG=DIR
100 UNTIL A$=«X»:REM PRESSER X POUR TERMINER
110 ?DATREG=0:REM MISE HORS TENSION
```

COMMODORE 64

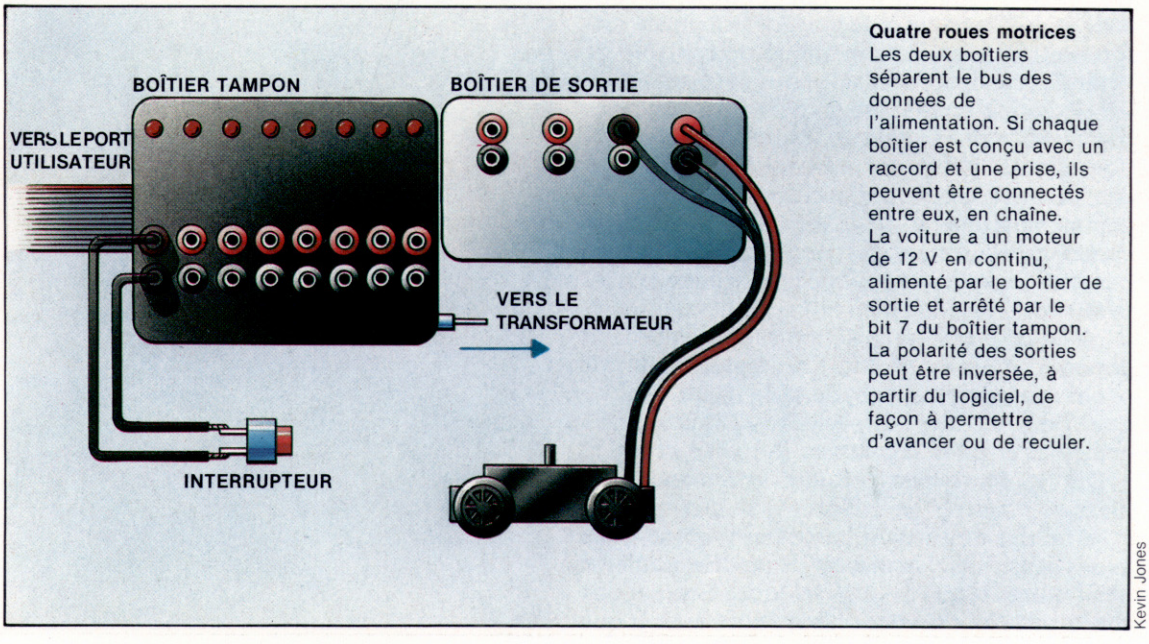
```
10 REM MOTEURS DIRIGÉS CBM 64
20 DDR=56579:DATREG=56577
30 POKEDDR,127:REM ENTRÉE LIGNE 7
40 POKEDATREG,0:REM TOUS HORS TENSION
50 GETA$:IFA$=«» THEN 50:REM ATTENDRE LA FRAPPE D'UNE
   TOUCHE
60 GETA$
70 IF (PEEK(DATREG)AND128)=0 THEN POKEDATREG, 2:GOTO90
80 POKEDATREG,1
90 IFAS<>«X»THEN60
100 POKEDATREG,0:REM MISE HORS TENSION
```

BBC MICRO

```
10 REM COMMANDE DE MOTEUR VARIABLE BBC
20 DDR=&FE62:DATREG=&FE60:SPEED=30
30 ?DDR=255:REM TOUTES LIGNES SORTIE
40 ?DATREG=0:REM TOUTES HORS TENSION
50 A$=GET$;REM ATTENDRE UNE TOUCHE
60 REPEAT
70 A$=INKEY$(1)
80 ?DATREG=0:REM MISE HORS TENSION
90 FORI=1TO(100-SPEED):NEXT:REM TEMPORISATION 1
100 ?DATREG=1:REM MISE SOUS TENSION
110 FORI=1TO SPEED:NEXT:REM TEMPORISATION
120 IF A$=«D»THEN SPEED=SPEED-5
130 IF A$=«Z»THEN SPEED=SPEED+5
140 UNTIL A$=«X»
150 ?DATREG=0:REM MISE HORS TENSION
```

COMMODORE 64

```
10 REM COMMANDE DE MOTEUR VARIABLE CBM 64
20 DDR=56579:DATREG=56577:SPEED=30
30 POKEDDR,255:REM TOUTES LIGNES SORTIE
40 POKEDATREG,0:REM MOTEUR HORS TENSION
50 GETA$:IFA$=«»THEN 50:REM ATTENDRE UNE TOUCHE
60 GETA$
70 POKEDATREG=0:REM MISE HORS TENSION
80 FOR I=1 TO(100-SPEED):NEXT:REM TEMPORISATION
90 POKEDATREG=1:REM MISE SOUS TENSION
100 FOR I=1 TO SPEED:NEXT:REM TEMPORISATION 2
110 IF A$=«D»THEN SPEED=SPEED-5
120 IF A$=«Z»THEN SPEED=SPEED+5
130 IF A$<>«X»THEN 60
140 POKEDATREG,0:REM MISE HORS TENSION
```



Quatre roues motrices
Les deux boîtiers séparent le bus des données de l'alimentation. Si chaque boîtier est conçu avec un raccord et une prise, ils peuvent être connectés entre eux, en chaîne. La voiture a un moteur de 12 V en continu, alimenté par le boîtier de sortie et arrêté par le bit 7 du boîtier tampon. La polarité des sorties peut être inversée, à partir du logiciel, de façon à permettre d'avancer ou de reculer.

Kevin Jones

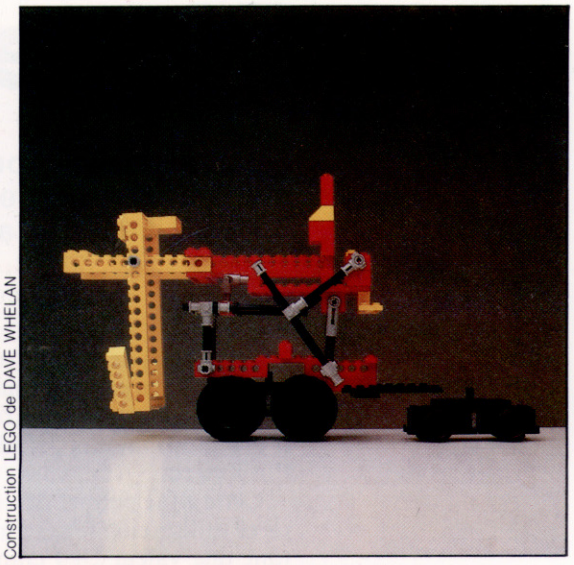
Dans ce programme, la variable SPEED sert à spécifier la longueur de chaque boucle de temporisation. Le code de la boucle est tel que, lorsqu'une temporisation diminue, l'autre augmente, et vice versa.

La temporisation 1 détermine la période pendant laquelle le moteur est hors tension, et la temporisation 2 la période pendant laquelle le moteur est sous tension.

Pour les grandes valeurs de SPEED, la première temporisation est courte tandis que la seconde est longue, ce qui fait tourner le moteur plus rapidement. De plus petites valeurs de SPEED allongeront les périodes de mise hors tension du moteur lors de chaque cycle, ce qui ralentira sa vitesse.

L'effet de pulsation que ce programme produit peut être observé au moyen du clignotement de la DEL de la ligne 1.

Tout va bien jusqu'ici
Piloter des voitures miniatures guidées par des câbles peut ne pas sembler représenter une grande récompense après tout l'effort investi dans ce travail ! Mais la construction et la programmation de ce système nous ont fait découvrir la construction électromécanique, et certains problèmes, certaines possibilités impliqués lorsque l'on crée un logiciel dont le but est de relier l'ordinateur au monde réel. (Cl. Ian McKinnell.)



Construction LEGO de DAVE WHELAN

Réponses aux exercices

1) En supposant que les capteurs sont connectés aux lignes 6 et 7 et que le moteur est connecté entre les bornes positives des lignes 0 et 1 :

```
10 REM BBC VERSION 3.1
20 DDR=&FE62:DATREG=&FE60
30 ?DDR=63:REM ENTREE DES LIGNES 6 ET 7
40 avant = 1:arrière=2
50 ?DATREG=avant
60 FORI=1TO2000:NEXT:REM TEMPORISATION
70 REPEAT UNTIL (?DATREG AND192)   192
80 ?DATREG=arrière
90 FORI=1TO2000:NEXT:REM TEMPORISATION
100 IF <?DATREG AND192)<> 192 THEN50
    ELSE GOTO 100
```

```
10 REM CBM 64 VERSION 3.1
20 DDR=56579:DATREG=56577
30 POKEDDR,63:REM ENTREE DES LIGNES 6 ET 7
40 FW=1:RV=2
50 POKE DATREG,FW
60 FORI=1TO1000:NEXT:REM TEMPORISATION
70 IF (PEEK (DATREG ) AND192)=192THEN70
80 POKE DATREG,RV
90 FORI=1TO1000:NEXT:REM TEMPORISATION
100 IF (PEEK(DATREG)AND192)<> 192THEN50
110 GOTO100
```

3) En supposant que les capteurs de 40 et de 70 degrés sont connectés aux lignes 6 et 7 respectivement et que le chauffe-eau est connecté à la ligne 0 :

```
10 REM BBC VERSION 3.3
20 DDR=&FE62:DATREG=&FE60
30 ?DDR=63:REM ENTREE DES LIGNES 6 ET 7
40 REPEAT
50 A$=INKEY$(1)
60 ?DATREG=1:REM METTRE LE CHAUFFE-EAU SOUS TENSION
70 REPEAT
80 UNTIL(?DATREG AND 192)=0:REM 70 DEG
90 ?DATREG=0:REM METTRE LE CHAUFFE-EAU HORS TENSION
100 REPEAT UNTIL(?DATREG AND192)=192
110 UNTIL A$<>"":REM UNE TOUCHE PRESSEE
```

```
10 REM CBM 64 VERSION 3.3
20 DDR=56579:DATREG=56577
30 POKE DDR,63:REM ENTREE DES LIGNES 6 ET 7
40 GET A$
50 ?DATREG=1:REM METTRE LE CHAUFFE-EAU SOUS TENSION
60 IF(PEEK(DATREG)AND192)<> 0THEN60
70 POKE DATREG,0:REM METTRE CHAUFFE-EAU HORS TENSION
80 IF(PEEK(DATREG)AND192)<> 192THEN80
80 IF A$=""THEN40
```

2) En supposant que le capteur est connecté à la ligne 7 et que le moteur est connecté entre les lignes 0 et 1 :

```
10 REM BBC VERSION 3.2
20 DDR=&FE62:DATREG=&FE60
30 ?DDR=127:REM ENTREE LIGNE 7
40 vitesse=30:avant =1:arrière=2
50 direction = avant
60 REPEAT
70 ?DATREG=0:REM OFF
80 FOR I=1 TO (100-vitesse):NEXT
90 ?DATREG=direction
100 FOR I=1TO1000:NEXT:REM
110 UNTIL(?DATREG AND128)=0:REM INTERRUPTEUR
120 FORI=1TO1000:NEXT:REM TEMPORISATION
130 TEST POUR CHEVAUCHEMENT DU CAPTEUR
140 IF(?DATREG AND128)=0THEN?DATREG=0:
    END
150 REM ARRIERE LENTEMENT
160 vitesse=2:direction=arrière=GOTO 60
```

```
10 REM CBM 64 VERSION 3.2
20 DDR=56579:DATREG=56577
30 POKE DDR,127:REM ENTREE LIGNE 7
40 SP=30:FW=1:RV=2
50 DR=FW
60 POKE DATREG,0: REM OFF
70 FORI=1TO(100-SP):NEXT
80 POKE DATREG,DR
90 FORI=1TOSP:NEXT
100 IF(PEEK(DATREG)AND128)<> 0THEN60
110 FORI=1TO1000:NEXT:REM TEMPORISATION
120 IF(PEEK(DATREG)AND128)=0THENPOKE
    DATREG,0:END
130 REM RECULER LENTEMENT
140 SP=2:DR=RV:GOTO60
```

4) En supposant que le premier interrupteur est sur la ligne 6 et que le deuxième est sur la ligne 7 :

```
10 REM BBC VERSION 3.4
20 DDR=&FE62:DATREG=&FE60:DISTANCE=1
30 ?DDR=63
40 REPEAT UNTIL(?DATREG AND64)=0
50 ?DATREG=1
60 TIME=0:REM DEMARRER MINUTERIE
70 REPEAT UNTIL(?DATREG AND128)=0
80 PRINT"VITESSE="DISTANCE/(TIME/100)
90 ?DATREG=0
```

```
10 REM CBM 64 VERSION 3.4
20 DDR=56579:DATREG=56577:DS=1
30 POKE DDR,63
40 IF(PEEK(DATREG)AND64)<> 0THEN40
50 POKE DATREG,1
60 T=TI:REM DEMARRER MINUTERIE
70 IF(PEEK(DATREG)AND128)<> 0THEN60
80 PRINT "VITESSE="DS/((TI-T)/60)
90 POKE DATREG,0
```

Former des caractères

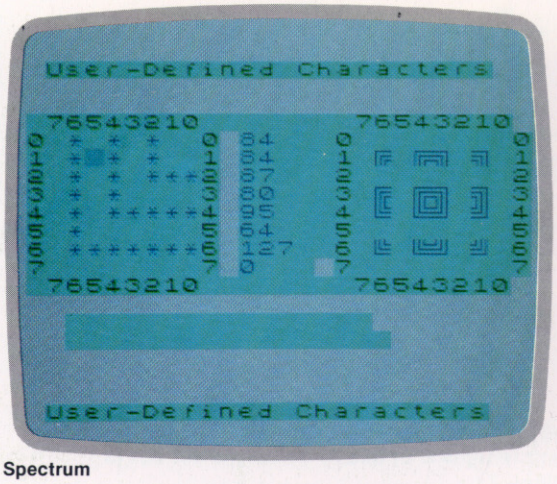
Nous avons vu comment définir ses propres caractères sur le Commodore. Nous donnons ici deux programmes destinés au BBC Micro et au Spectrum.

L'approche par le Spectrum des caractères utilisateur est très pragmatique : 168 octets sur le dessus de la mémoire sont réservés, par le système d'exploitation, aux définitions de caractères par l'utilisateur.

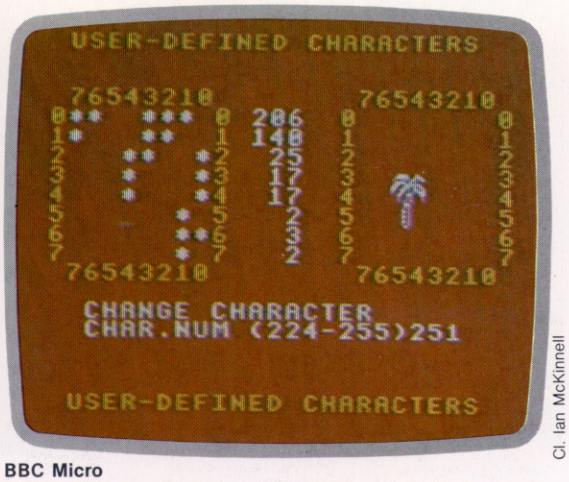
Cette zone permet de définir vingt et un caractères utilisateur, le système d'exploitation ajoutant une valeur CHR$ comprise entre 144 et 164 pour chaque définition. Les caractères définis sont considérés par la machine comme étant compris entre « a » et « u » en mode graphique (G). La variable système du graphisme défini par l'utilisateur (GDU) (adresse 32600) pointe sur le premier octet de la zone mémoire réservée, mais vous n'avez pas besoin d'effectuer des additions compliquées avec cette variable pour trouver l'adresse de début d'une définition de caractère. La commande LET adresse = USR«A» fournit l'adresse du premier octet de la définition du caractère entre guillemets.

La gestion faite par le BBC Micro du graphisme défini utilisateur est semblable à première vue à celle du Spectrum. Les 256 octets compris entre &COOO et &OCOO sont réservés aux définitions des caractères codés ASCII compris entre 224 et 225. Si le jeu de caractères est « condensé », ces définitions concernent également les codes ASCII compris entre 128 et 159, 160 et 191, 192 et 223. Si le jeu de caractères est « ventilé », le jeu entier de caractères imprimables (depuis CHR$(32) jusqu'à CHR$(255)) peut être redéfini, mais au prix d'espace de RAM (mémoire vive). Les fonctions du programme sont les mêmes que pour les versions du Commodore et du Spectrum.

Les touches fléchées contrôlent les déplacements du curseur. Les touches de fonctions f1 à f3 permettent d'accéder aux commandes bascule, redéfinition et placer. Le point d'exclamation (!) est le terminateur du programme.



Spectrum



Cl. Ian McKinnell

BBC Micro

Le programme est la traduction littérale de la version du Commodore. Les touches normales de contrôle du curseur du Spectrum (touche préfixée 5, touche préfixée 6, etc.) dirigent le curseur d'édition, et les touches 6, 7 et 8 non préfixées sont les touches de commandes : bascule d'une position, changement du caractère d'édition et placement d'un caractère dans la fenêtre texte. Le point d'exclamation est la commande de sortie. Une fois vos huit caractères définis, vous pouvez sauvegarder la zone du graphisme défini utilisateur (GDU) par :

SAVE«fichierGDU» CODE(USR«A»), 168 (USR = UtiliSateuR),

et la recharger de la sorte :

LOAD«fichierGDU» CODE

Les appels OS aux lignes 70 et 180 activent et inhibent les fonctions COPY pour permettre l'utilisation des touches fléchées dans le programme. Si vous quittez le programme d'une autre manière que par la commande «!», il vous faudra taper « *FX4,0 » pour réactiver les touches d'édition.

Le BBC Micro vous permet d'utiliser VDU23 pour définir des caractères, mais il est plus simple alors de calculer les adresses appropriées et de les soumettre séparément à PEEK et à POKE.

Vous pouvez sauvegarder vos caractères spéciaux de la sorte :

*SAVE «nomfichier» OCOO OCFF

et le recharger de la sorte :

*LOAD«»

Spectrum

```
  19 REM **********************
  20 REM *       SPECTRUM       *
  21 REM *    GEN. CAR. LIT     *
  22 REM **********************
 100 GOSUB 1000: REM initialiser
 110 FOR j=0 TO 1 STEP 0
 120 GOSUB 2500: REM saisie
 130 GOSUB 3000: REM valider
 140 GOSUB obéir: REM pointeur
 150 NEXT j
 200 STOP
 999 REM **********************
1000 REM *     initialiser      *
1001 REM **********************
1020 DIM b(8,8):  DIM oui(2,2):
  DIM o(2,7): DIM i$(8): DIM d$(
1): DIM t(8,8)
1100 LET touchenulle=2000: LET
dptcrsr=3500: LET commande=3000: LET
miseàjour=6500
1200  REM ****init écran****
1220 LET i0=4: LET c0=3: LET i1=
8: LET c1=8: LET of=16
1230 LET noir=0: LET bleu=1: LE
T cyan=5: LET blanc=7
1240 PAPER cyan: INK noir
1250 LET z$="caractères utilisateur":
PRINT AT 1,4;z$:
 20,4;z$
1260 LET z$=" 76543210":
  i0,c0;z$;: PRINT TAB c0+of;z$
1270 PRINT AT i0+c1+1,c0;z$;: PR
INT TAB c0+of;z$
1280 FOR i=i0+1 TO i0+c1
1290 LET z$=CH$ (i-i0-1): LET z
$=z$+"          "+z$
1300 PRINT AT i,c0;z$;AT i,c0+of
;z$
1310 NEXT i
1420 REM ****valeur dpt curs.****
1440 DATA -1,+1,0,0
1445 DATA 0,0,+1,-1
1450 DATA 0,0,+1,-1
1460 FOR k=1 TO 2: FOR l=1 TO 4
1470 READ o(k,l): NEXT l: NEXT k
1600 REM ***init. fenêtre texte***
1620 FOR i=1 TO i1
1640 FOR c=1 TO c1
1660 LET t(i,c)=32
1680 NEXT c: NEXT i
1800 LET ipos=1: LET cpos=1: LET
 cn=144: LET e$="A"
1820 GO SUB 6000
1990 RETURN
1999 REM **********************
2000 REM *  frappe illégale    *
2001 REM **********************
2020 BEEP .2,-3: RETURN
2120 RETURN
2499 REM **********************
2500 REM *  crsr @ipos, cpos *
2501 REM **********************
2520 LET ip=i0+ipos: LET  cp=c0+
cpos: LET cf=1+b(ipos,cpos): LET
 d$=" *"(cf)
2540 PRINT AT ip,cp; FLASH 1;d$
2600 REM  LET t$=SAISIE$: IF t$<>
"" THEN  GOTO 2600
2620 LET t$=SAISIE$: IF t$="" THE
N  GOTO 2620
2640 PRINT AT ip,cp; FLASH 0;d$
2700 RETURN
2999 REM **********************
3000 REM * valider la saisie *
3001 REM **********************
3020 IF t$="!" THEN LET pointeur
=touchenulle: LET j=2: RETURN
3040 LET touche=CODE (t$)-7: LET k=
touche-45: LET pointeur=touchenulle
3060 IF (touche)=1 AND touche<=4) THEN
  LET pointeur=dptcnsr: RETURN
3080 IF (t$)="5" AND t$<="8") TH
EN  LET touche=VAL (t$)-4 : LET
pointeur=commande+touche*500: RETUR
N
3100 RETURN
3499 REM **********************
3500 REM * déplacer le curseur *
3501 REM **********************
3520 LET ny=ipos+o(2,touche):  LET
nx=cpos+o(1,touche)
3540 IF  (ny<1 OR ny>i1)  THEN
  RETURN
3560 IF  (nx<1 OR nx>i1)  THEN
  RETURN
3580 LET ipos=ny: LET cpos=nx
3600 RETURN
3999 REM **********************
4000 REM * bascule d'une cellule *
4001 REM **********************
4020 LET bs=1-b(ipos,cpos): LET
d$=" *"(1+bs)
4040 PRINT AT ipos+i0,cpos+c0;d$
4060 LET b(ipos,cpos)=bs
4120 LET pe=PEEK (mpos+ipos)
4140 LET pe=pe+(bs*2-1)*(2^(8-cp
os))
4160 POKE (mpos+ipos),pe
4200 PRINT AT i0+ipos,c0+c1+3;PE
EK (mpos+ipos);" "
4220  NEXT i
4300 RETURN
4499 REM **********************
4500 REM *  déf. nouveau car. *
4501 REM **********************
4520 PRINT  AT i0+i1+3,c0+2;
 FLASH 1;"changer car."
4530 GOSUB mise à jour
4540 PRINT  AT i0+i1+4,c0+2;
;"frappez une touche (a - u)"
4550 FOR z=1 TO 1
4560 LET e$=saisie$: IF e$<>""
 THEN GOTO 4560
4570 LET e$=saisie$: IF e$=""
THEN  GOTO 4570
4580 IF (e$<"a" OR e$>"u") THEN
 GOSUB touchenulle: LET z=0
4590 NEXT z
4600 LET cn=CODE (e$)+79
4610 IF cn>164 THEN  LET cn=cn-3
2
4620 PRINT  AT i0+i1+3,c0+2;
 FLASH 0;"                "
4630 PRINT  AT i0+i1+4,c0+2;
"                "
4640 GOSUB 6000
4990 RETURN
4999 REM **********************
5000 REM *    placer un car.    *
5001 REM **********************
5020 PRINT  AT ipos+i0,cpos+c0+
of;CAR$ (cn)
5040 LET t(ipos,cpos)=cn
5490 RETURN
5999 REM **********************
6000 REM *   afficher un car.  *
6001 REM **********************
6020 LET mpos=UTS (e$)-1:
 INK bleu
6040 FOR i=1 TO c1: LET pe=PEEK
(mpos+i): LET p=pe: LET z$=""
6060 FOR c=c1 TO 1 STEP -1: LET
n=INT (pe/2): LET q=pe-2*n
6080 LET b(i,c)=q: LET pe=n
6100 LET i$(c)=" *"(q+1): NEXT c
6120 PRINT AT i0+i,c0+1;i$;AT r0
+i,c0+c1+3;p;"  "
6130 NEXT i
6140 RETURN
6499 REM **********************
6500 REM *   MAJ fen. texte    *
6501 REM **********************
6520 FOR i=" TO 8: LET y=i0+i
6540 FOR c=1 TO 8: LET x=c0+c+of
6560 PRINT  AT y,x;CAR$ (t(r,c))
6580 NEXT c: NEXT i
6600 RETURN
```

BBC Micro

```
  19 REM************************
  20 REM*       BBC Micro       *
  21 REM************************
  22 REM* GEN CAR. UT.          *
  50 MODE 1
  60 Q%=3
  70 *FX4,1
 100 PROCinitialisation
 110 REPEAT
 120 PROCget_commande
 130 PROCvalidation commande
 140 PROCobéir(COMMANDE)
 160 UNTIL COMMANDE=SORTIR
 180 *FX4,0
 200 END
 999 REM************************
1000 DEFPROCinitialisation
1001 REM************************
1040 DIM BD (8,8) ,C$ (2,2)
1045 DIMOFST (2,7) ,D$(1),TX(8,8)
1060 D$(0)=" ":D$(1)="*"
1080 CGEN=&0C00-1
1100 SORTIR=-1:TOUCHENULLE=0:DPTCRSR=1:
1105 BS=2:DFINIR=3:PLACER=4
1200 REM*****INIT ECRAN*****
1220 i0=4:C0=3:iL=8:CL=8
1225 C9=CL+3:OF=16
1230 NOIR=0:ROUGE=1:JAUNE=2
1235 BLANC=3:FOND=128
1240 COLOUR FOND+ROUGE
1245 COLOUR JAUNE:CLS
1250 Z$="CARACTERES UTILISATEUR"
1255 PRINTTAB(4,1)Z$;TAB(4,20)Z$
1260 Z$=" 76543210"
1264 PRINTTAB(C0,R0)Z$
1268 PRINTTAB(C0+OF,R0)Z$
1270 Y=i0+iL+1:X=C0
1274 PRINTTAB(X,Y)Z$
1278 PRINTTAB(X+OF,Y)Z$
1280 FOR i=i0+1 TO i0+iL
1290 Z$=CH$(R-R0-1)
1295 Z$=Z$+"        "+Z$
1300 PRINTTAB(C0,i)Z$
1304 PRINTTAB(C0+OF,i)Z$
1310 NEXT i
1320 COLOUR BLANC
1420 REM****VAL. DPT CRSR****
1440 DATA -1,+1,0,0
1450 DATA 0,0,+1,-1
1460 FOR K=1 TO 2:FOR L=1 TO 4
1470 READ VAL.DPT(K,L):NEXT L,K
1500 REM****ATTRIBUTION DES TOUCHES****
1520 *KEY1"T"
1540 *KEY2"D"
1560 *KEY3"P"
1600 REM**INIT. FEN. TEXTE**
1620 FOR i=1 TO iL
1640 FOR C=1 RO iL
1660 TX(i,C)=32
1680 NEXT C,i
1800 iPOS=1:CPOS=1:CN=224
1805 PROC Aff car
1990 ENDPROC
1999 REM************************
2000 DEFPROCtouchenulle
2001K REM************************
2020 SOUND 1,-15,48,10
2120 ENDPROC
2499 REM************************
2500 DEFPROCprendre-com
2501 REM************************
2520 iP=i0+iPOS:CP=C0+CPOS
2540 PRINTTAB (CP,iP) ;
2620 T$=GET$
2700 ENDPROC
2999 REM************************
3000 DEFPROCvalider-com.
3001 REM************************
3020 COMMANDE=DPTCRSR
3040 KEY=ASC (T$) -135
3060 IF KEY<1 THEN COMMANDE=TOUCHENULLE
3065 IF KEY>4 THEN COMMANDE=TOUCHENULLE
3080 IF T$="T"THEN COMMANDE=BASCULE
3100 IF T$="D"THEN COMMANDE=DFINIR
3120 IF T$="P"THEN COMMANDE=PLACER
3150 IF T$="!"THEN COMMANDE=SORTIR
3160 ENDPROC
3299 REM************************
3300 DEFPROCobéir(commande)
3301 REM************************
3320 IF commande=SORTIR THEN
PROCtouchenulle
3340 IF commande=OPTCRSR
PROCdéplacercurseur(TOUCHE)
3360 IF commande=BS THEN
PROCbasculer_une_cellule
3380 IF commande=DFINIR THEN
PROCdéfinir_un_caractère
3400 IF commande=PLACER THEN
PROCplacer un caractère
3420 IF touchenulle THEN
PROCtouchenulle
3450 ENDPROC
3499 REM************************************
3500 DEFPROCdéplacementcurseur(touche)
3501 REM************************************
3520 NY=iPOS+OFST(2,touche)
3525 NX=CPOS+OFST(1,touche)
3540 IF NY>=1 AND NY<=RL THEN
RPOS=NY ELSE PROCtouchenulle
3560 IF NX>=1 AND NX<=CL THEN
CPOS=NX ELSE PROCtouchenulle
3600 ENDPROC
3999 REM************************
4000 DEFPROCbasculer_une_cellule
4001 REM************************
4020 BS=1-BD<iPOS,CPOS>
4025 Y=iPOS+R0:X=CPOS+C0
4040 PRINTTAB(X,Y)D$(BS)
4060 BD iPOS,CPOS=BS
4120 PE=?(MPOS+iPOS)
4140 PE=PE+(BS*2-1)*(2 (B-CPOS))
4160 ?(MPOS+iPOS)=PE
4200 PRINTTAB C0+C9,i0+iPOS)PE
4220 PROCmise_à_jour_texte
4300 ENDPROC
4499 REM************************
4500 DEFPROCdéfinir_caractère
4501 REM************************
4510 REPEAT
4520 PRINTTAB(C0+2,i0+iL+3);
4525 PRINT"                "
4540 PRINTTAB(C0+2,i0+iL+4);
4545 PRINT"                "
4550 PRINTTAB(C0+2,i0+iL+4);
4560 INPUT"NOMBRE(224-255)"CHNOM
4580 UNTIL CHNOM223 AND CHNOM 256
4600 CN=CHNOM
4620 PROCaffichase_d'un_caractère
4900 ENDPROC
4999 REM************************
5000 DEFPROCplacer_un_caractère
5001 REM************************
5015 X=CPOS+C0+OF:Y=iPOS+i0
5020 PRINTTAB(X,Y);CAR$(CN)
5040 TX(CPOS,iPOS)=CN
5400 ENDPROC
5999 REM************************
6000 DEFPROCafficher_caractère
6001 REM************************
6020 MPOS= (CN-224)*8+CGEN
6040 FOR i=1 TO CL
6045 PE=?(MPOS+i):P=PE:Z$=""
6060 FOR C=CL TO 1 STEP -1
6065 N=INT(PE/2):Q=PE-2*N
6080 BD(R,C)=Q:PE=N
6100 Z$=D$(Q)+Z$:NEXT C
6120 PRINTTAB (C0+1,i0+i)Z$
6125 PRINTTAB (C0+C9,i0+i)P
6130 NEXT i
6140 ENDPROC
6499 REM************************
6500 DEFPROCmise_à_jour_texte
6501 REM************************
6520 FOR i=1 TO iL:Y=i0+i
6540 FOR C=1 TO iL:X=C0+C+OF
6560 PRINTTAB(X,Y)CAR$(TX(C,i))
6580 NEXT C,i
6900 ENDPROC
```

N.B. Le i signifie ligne et remplace ici le r de l'anglais row.

Au doigt et à l'œil

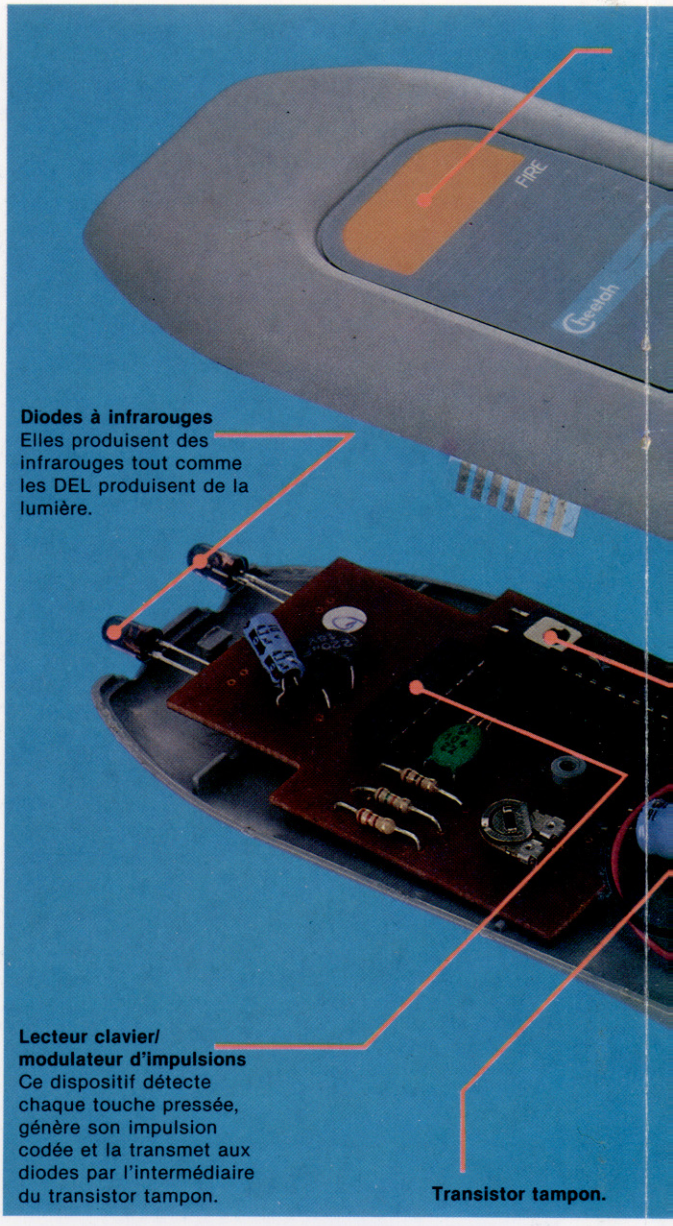
Cheetah Marketing a déjà réalisé bien des extensions destinées au Spectrum; sa plus grande réussite reste sans doute le synthétiseur de parole Sweet Talker. Elle vient tout juste de mettre sur le marché le RAT, un manche à balai qui fonctionne à distance, grâce aux infrarouges.

Les passionnés de jeux d'arcades ne négligent aucun détail qui peut améliorer, si peu que ce soit, la qualité et la rapidité de leur jeu, surtout si leur score final s'en trouve augmenté. C'est bien pourquoi la façon dont on guide le déroulement d'une partie a tant d'importance. Si c'est à partir du clavier, il faudra que les touches de contrôle soient bien réparties et faciles à utiliser. C'est même un aspect auquel les auteurs de logiciels se doivent d'être particulièrement attentifs. Mais un manche à balai, qui est un dispositif de commande extérieur à l'ordinateur, est à envisager du point de vue du matériel.

Ce qui importe avant tout, c'est bien évidemment sa souplesse d'emploi : liberté de mouvement, rapidité de réaction, etc. Mais la nature même de l'objet lui impose certaines limites : longueur du câble, taille, forme et emplacement de la poignée de commande, position du (ou des) bouton(s) de mise à feu. Ce dernier point, par exemple, favorise souvent les droitiers. Les constructeurs ont depuis longtemps cherché à surmonter ces handicaps, mais aucun ne semble être parvenu à réaliser un engin aussi réussi que le manche à balai à infrarouges que Cheetah Marketing a réalisé pour le Spectrum.

L'objet a reçu le nom de RAT, ce qui veut dire, paraît-il, *remote action transmitter* (transmetteur d'action à distance), mais sans doute est-ce plutôt un jeu de mots sur *mouse* (souris), le fameux procédé de guidage dont on se sert avec le Macintosh d'Apple et avec plusieurs ordinateurs du même genre. Le RAT a quelque chose d'un peu futuriste. Il est long, gris et plat, avec un dispositif de contrôle circulaire bleu, un bouton de mise à feu orange vif; il comporte aussi le logo de Cheetah. Deux transmetteurs infrarouges dépassent à l'avant. La première fois que vous l'avez entre les mains, vous avez l'impression qu'il va en jaillir des flammes bleues.

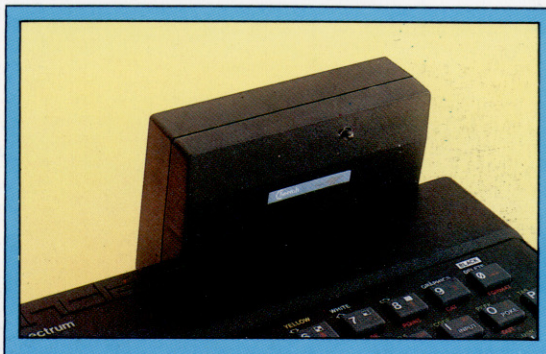
Le système dispose également de sa propre interface, qui se raccorde au connecteur plat situé



Diodes à infrarouges
Elles produisent des infrarouges tout comme les DEL produisent de la lumière.

Lecteur clavier/ modulateur d'impulsions
Ce dispositif détecte chaque touche pressée, génère son impulsion codée et la transmet aux diodes par l'intermédiaire du transistor tampon.

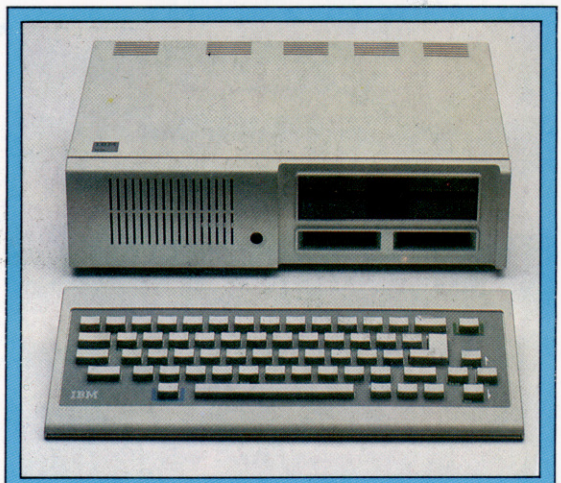
Transistor tampon.



Chris Stevens

Que la lumière soit
Les rayons infrarouges sont émis par le transmetteur grâce à des diodes; un courant électrique passe dans une puce d'arsénure de gallium, dont les molécules sont excitées provoquant l'émission de photons. Dans le récepteur, c'est l'inverse qui se passe : le courant passe dans la diode quand les rayons atteignent l'arsénure. Lorsque le dispositif de contrôle du RAT enregistre des pressions, les deux diodes du transmetteur émettent des impulsions infrarouges codées, en un large faisceau, qui atteint le récepteur, directement ou après réflexion dans la pièce.

Chris Stevens



Des réactions mitigées
Lorsque le PC Junior d'IBM fit enfin son apparition, on a surtout noté la médiocrité de conception et de réalisation de son clavier. C'était pourtant le premier micro-ordinateur qui ait relié le microprocesseur et le clavier par un système à infrarouges.

à l'arrière du Spectrum. Cette petite boîte est, de surcroît, équipée d'un port d'expansion, en prévision d'autres périphériques. Elle n'a qu'un seul récepteur à infrarouges, installé à l'avant, pour communiquer avec le RAT.

Une petite brochure vous indique comment vous servir de l'appareil, et précise quels sont les jeux avec lesquels en faire usage (tout programme compatible avec le manche à balai de type Kempston). Cheetah, qui voit loin, a même pris la peine d'inclure dans le texte des routines en BASIC et en langage machine, de sorte que le RAT puisse être mis en œuvre avec vos propres jeux.

Les mêmes instructions précisent que le système fonctionne jusqu'à 4 m de distance et qu'il suffit de le pointer « en direction de l'ordinateur ». On commande un mouvement en appuyant légèrement sur le dispositif de contrôle. Il comporte huit petites « encoches » à sa périphérie; en appuyant sur l'une d'elles (ou juste à côté), vous sélectionnez une direction particulière (nord, sud-ouest, etc.); bref, le tout est un peu comparable à une rose des vents. Tandis qu'une main tient l'appareil et règle les déplacements, l'autre est posée sur le bouton de mise à feu. Vu la façon dont le RAT a été dessiné, les deux mains peuvent accomplir indifféremment les deux tâches, et les gauchers ne sont donc plus désavantagés.

Le tout doit être équipé d'une pile, qui est placée dans une petite cavité juste au-dessous du dispositif de contrôle; il suffit de charger un jeu pour démarrer. Aucun signe ne vous permet de savoir si le transmetteur fonctionne bel et bien, aussi vous retrouverez-vous peut-être collé à l'écran... ce qui vous empêchera de profiter des avantages du RAT. Au début, vous vous croirez sans doute contraint de ne pas vous éloigner. Pourtant, une fois que vous serez certain que le système fonctionne, toutes les expériences seront possibles.

Le RAT donne d'excellents résultats, même à des distances légèrement supérieures à celles qu'indique le constructeur. Il n'est pas besoin non plus de le pointer en direction de l'ordinateur. Qu'il soit dirigé vers le plafond ou le plancher, manipulé dans le dos ou sur le côté, il transmet toutes vos commandes — mais il est alors un peu difficile de voir ce qu'on fait... Le joueur se voit donc offrir une liberté de déplacement beaucoup plus grande. Un gros inconvénient cependant : l'appareil n'indique que huit directions (haut, bas, droite, gauche et directions intermédiaires). Il aurait été encore plus intéressant s'il en avait comporté davantage.

Comme le RAT ne comporte aucune partie mobile, il est beaucoup moins exposé à la fatigue et au délabrement que connaissent tous les manches à balai standard, et devrait donc donner satisfaction très longtemps. Il est d'ailleurs garanti pour une durée de un an. Il est d'un prix raisonnable, à peine plus cher qu'un manche à balai standard plus l'Interface 2 (mais peut-être, il est vrai, avez-vous déjà une Interface 2). Il offre en tout cas beaucoup plus de liberté et de précision que les autres appareils de ce type.

Manche à balai RAT à infrarouges destiné au : Spectrum.

Fonctionne avec : tous les jeux acceptent les manches à balai Cheetah ou Kempston.

Constructeur : Cheetah Marketing.

Carte Z80

Voici reproduit, avec l'aimable autorisation de Zilog Inc., la première partie de la carte référence du programmeur Z80. Au-delà de sa complexité, il s'agit d'un outil important pour tous ceux qui veulent programmer.

ENS. REG. PRINCIPAL **ENS. REG. ALT.**

Accumulateur A	Drapeaux F	Accumulateur A′	Drapeaux F′
B	C	B′	C′
D	E	D′	E′
H	L	H′	L′

Registres généraux

Vecteur Interrupt. I	Rafraîch. mémoire R
Registre d'index IX	
Registre d'index IY	
Pointeur de pile SP	
Compteur de programme PC	

Registres spéciaux

Instruction	D_7 S	Z		H		P/V	N	D_0 C	Commentaires
ADD A, s ; ADC, s	↕	↕	X	↕	X	V	0	↕	Addition 8 bits ou addition avec retenue soustraction
SUB s ; SBC A, s ; CP s ; NEG	↕	↕	X	↕	X	V	1	↕	8 bits, soustraction avec retenue, comparaison et négation de l'accumulateur.
AND s	↕	↕	X	1	X	P	0	0	} Opérations logiques.
OR s, XOR s	↕	↕	X	0	X	P	0	0	}
INC s	↕	↕	X	↕	X	V	0	•	Incrémentation 8 bits.
DEC s	↕	↕	X	↕	X	V	1	•	Décrémentation 8 bits.
ADD DD, ss	•	•	X	X	X	•	0	↕	Addition 16 bits.
ADC HL, ss	↕	↕	X	X	X	V	0	↕	Addition 16 bits avec retenue.
SBC HL, ss	↕	↕	X	X	X	V	1	↕	Soustraction 16 bits avec retenue.
RLA, RLCA, RRA ; RRCA	•	•	X	0	X	•	0	↕	Rotation accumulateur.
RL m ; RLC m ; RR m ; RRC m ; SLA m ; SRA m ; SRL m	↕	↕	X	0	X	P	0	↕	Rotation et décalage d'emplacements.
RLD ; RRD	↕	↕	X	0	X	P	0	•	Rotation de chiffre à gauche et à droite.
DAA	↕	↕	X	↕	X	P	•	↕	Ajustement décimal de l'accumulateur.
CPL	•	•	X	1	X	•	1	•	Complément de l'accumulateur.
SCF	•	•	X	0	X	•	0	1	Retenue.
CCF	•	•	X	X	X	•	0	↕	Complément de retenue.
IN r (C)	↕	↕	X	0	X	P	0	•	Entrée de registre indirect.
INI ; IND ; OUTI ; OUTD	X	↕	X	X	X	X	1	•	} Entrée et sortie de bloc. Z = 0 si B = 0, sinon Z = 1.
INIR ; INDR ; OTIR ; OTDR	X	1	X	X	X	X	1	•	} Instructions de transfert de bloc. P/V = 1 si BC = 0,
LDI ; LDD	X	X	X	0	X	↕	0	•	} sinon P/V = 0.
LDIR ; LDDR	X	X	X	0	X	0	0	•	}
CPI ; CPIR ; CPD ; CPDR	X	↕	X	X	X	↕	1	•	Instructions de recherche de bloc. Z = 1 si A = (HL), sinon Z = 0. P/V = 1 si BC = 0, sinon P/V = 0.
LD A, I, LD A, R	↕	↕	X	0	X	IFF	0	•	Le contenu de la bascule d'interruption (interrupt. flip-flop/IFF) est copié dans le drapeau P/V.
BIT b, s	X	↕	X	1	X	X	0	•	L'état de bit b d'emplacement est copié dans le drapeau Z.

Mnémonique	Opération symbolique	Drapeaux S	Z	H	P/V	N	C	Opc 76	543	210	hex	Nombre d'octets	Nombre de cycles	Nombre d'états T	Commentaires r, r′	Reg.
LD r, r	r ← r	•	•	X	•	X	•	•	•	01	r	r		1	1	4	000	B
LD r, n	r ← n	•	•	X	•	X	•	•	•	00	r	110		2	2	7	001	C
											← n →						010	D
LD r, (HL)	r ← (HL)	•	•	X	•	X	•	•	•	01	r	110		1	2	7	011	E
LD r, (IX + d)	r ← (IX + d)	•	•	X	•	X	•	•	•	11	011	101	DD	3	5	19	100	H
										01	r	101					101	L
											← d →						111	A
LD r, (IY + d)	r ← (IY + d)	•	•	X	•	X	•	•	•	11	111	101	FD	3	5	19		
										01	r	110						
											← d →							
LD (HL), r	(HL) ← r	•	•	X	•	X	•	•	•	01	110	r		1	2	7		
LD (IX + d), r	(IX + d) ← r	•	•	X	•	X	•	•	•	11	011	101	DD	3	5	19		
										01	110	r						
											← d →							
LD (IY + d), r	(IY + d) ← r	•	•	X	•	X	•	•	•	11	111	101	FD	3	5	19		
										01	110	r						
											← d →							
LD (HL), n	(HL) ← n	•	•	X	•	X	•	•	•	00	110	110	36	2	3	10		
											← n →							
LD (IX + d), n	(IX + d) ← n	•	•	X	•	X	•	•	•	11	011	101	DD	4	5	19		
										00	110	110	36					
											← d →							
											← n →							
LD (IY + d), n	(IY + d) ← n	•	•	X	•	X	•	•	•	11	111	101	FD	4	5	19		
										00	110	110	36					
											← d →							
											← n →							
LD A, (BC)	A ← (BC)	•	•	X	•	X	•	•	•	00	001	010	0A	1	2	7		
LD A, (DE)	A ← (DE)	•	•	X	•	X	•	•	•	00	011	010	1A	1	2	7		
LD A, (nn)	A ← (nn)	•	•	X	•	X	•	•	•	00	111	010	3A	3	4	13		
											← n →							
											← n →							
LD (BC), A	(BC) ← A	•	•	X	•	X	•	•	•	00	000	010	02	1	2	7		
LD (DE), A	(DE) ← A	•	•	X	•	X	•	•	•	00	010	010	12	1	2	7		
LD (nn), A	(nn) ← A	•	•	X	•	X	•	•	•	00	110	010	32	3	4	13		
											← n →							
											← n →							
LD A, I	A ← I	↕	↕	X	0	X	IFF	0	•	11	101	101	ED	2	2	9		
										01	010	111	57					
LD A, R	A ← R	↕	↕	X	0	X	IFF	0	•	11	101	101	ED	2	2	9		
										01	011	111	5F					
LD I, A	I ← A	•	•	X	•	X	•	•	•	11	101	101	ED	2	2	9		
										01	000	111	47					
LD R, A	R ← A	•	•	X	•	X	•	•	•	11	101	101	ED	2	2	9		
										01	001	111	4F					

Symbole — Opération

- S — Drapeau de signe. S = 1 si le MSB du résultat est 1.
- Z — Drapeau de zéro. Z = 1 si le résultat de l'opération est nul.
- P/V — Drapeau de parité ou de dépassement. La parité (P) et le dépassement (V) partagent le même drapeau. Les opérations logiques affectent la parité, tandis que les opérations arithmétiques affectent le dépassement. Dans le premier cas, P/V = 1 si le résultat est pair, P/V = 0 si le résultat est impair. S'il s'agit de dépassement, P/V = 1 si le résultat de l'opération a produit un dépassement.
- H — Drapeau de demi-retenue. H = 1 si l'addition ou la soustraction a produit une retenue sur le bit 4 de l'accumulateur.
- N — Drapeau d'addition-soustraction. N = 1 si l'opération précédente était une soustraction.
- H & N — H et N sont utilisés conjointement à l'instruction d'ajustement décimal (DAA) pour corriger le résultat en format BCD suivant l'addition ou la soustraction utilisant des opérandes en format BCD.
- C — Drapeau de retenue/lien. C = 1 si l'opération a produit une retenue du MSB de l'opérande ou du résultat.
- ↕ — Le drapeau est affecté suivant le résultat de l'opération.
- • — Le drapeau n'est pas modifié par l'opération.
- 0 — Le drapeau est remis à zéro par l'opération.
- 1 — Le drapeau est mis à un par l'opération.
- X — Le drapeau est indifférent.
- V — Drapeau P/V affecté suivant le résultat de dépassement de l'opération.
- P/V — Drapeau P/V affecté suivant le résultat de parité de l'opération.
- t — L'un des registres d'UC : A, B, C, D, E, H ou L.
- s — N'importe quel emplacement 8 bits pour tous les modes d'adressage autorisés par cette instruction.
- ss — N'importe quel emplacement 16 bits pour tous les modes d'adressage autorisés par cette instruction.
- " — N'importe lequel des deux registres d'index IX ou IY.
- R — Compteur de rafraîchissement.
- n — Valeur à 8 bits comprise dans l'intervalle < 0,255 >.
- nn — Valeur à 16 bits comprise dans l'intervalle < 0,65535 >.

Notes :
r, r′ représentent n'importe lequel des registres A, B, C, D, E, H, L.
IFF, contenu de la bascule d'interruption, est copié dans le drapeau P/V.

Notation de drapeaux
• = drapeau non affecté ; 0 = drapeau à zéro ;
1 = drapeau mis ; X = drapeau inconnu ;
↕ = drapeau affecté selon le résultat de l'opération.

Variations sur une tortue

Notre étude du langage LOGO nous a appris comment définir des procédures pour effectuer des séquences de commandes. Voyons comment obtenir plusieurs effets avec la même procédure.

Avec LOGO, un mot, par exemple «taille», peut être utilisé de trois manières. Afin de les distinguer, LOGO utilise trois notations : TAILLE, :TAILLE (que l'on dit « taille apostrophe »), «TAILLE (que l'on dit « taille guillemets »).

Lorsque LOGO rencontre le mot TAILLE non précédé d'une ponctuation, il le considère comme un nom de procédure et effectue la séquence de commandes présente dans la définition de TAILLE.

:TAILLE indique la valeur contenue dans le nom de la variable. Lorsque LOGO rencontre :TAILLE, il restitue la valeur associée au nom.

«TAILLE fait référence aux noms de variables et de procédures. Ainsi «TAILLE est utilisé pour faire référence à une variable, et :TAILLE renvoie à la valeur attribuée à la variable. Vous remarquez qu'il n'y a pas de guillemets après le nom, la fin du nom étant signalée par un blanc.

Le LOGO du MIT (Massachusetts Institute of Technology) ne coïncide pas toujours dans son utilisation avec cette notation. Après les commandes ÉDIT, EFFACE et PO, un nom de procédure ne doit pas être précédé de guillemets. Ainsi, une syntaxe exacte est ÉDIT CARRÉ, même si CARRÉ n'est pas un appel à la procédure CARRÉ, mais qu'elle représente simplement le nom de la procédure. Le LOGO de LCSI est plus logique et respecte la règle des guillemets, même avec ces commandes.

Pour utiliser une des procédures que nous avons définies jusqu'ici, il suffit de taper son nom, exactement de la même manière que pour des commandes LOGO comme DESSINE ou CACHE TORTUE. Cependant, d'autres commandes, telles qu'AVANCE, supposent des informations supplémentaires pour être utilisées.

Le mot AVANCE tout seul est dépourvu de signification, et une valeur doit lui être associée pour que LOGO puisse exécuter la commande. Si les procédures comprenaient des noms de variables, il serait possible de passer les valeurs voulues et de varier ainsi les effets obtenus à l'appel des procédures.

Reprenons la procédure que nous avons définie dans un article précédent pour tracer un carré :

```
POUR CARRÉ
      RÉPÈTE 4[AV 50 DR 90]
FIN
```

Cette procédure tracera un carré de 50 unités de côté. Elle serait cependant beaucoup plus utile si elle pouvait dessiner un carré dont on pourrait choisir les dimensions. Il suffirait alors d'entrer au clavier la valeur retenue. Pour modifier CARRÉ afin qu'elle accepte cette saisie, utilisez l'éditeur pour remplacer la valeur figée 50 par la variable «CÔTÉ et ajoutez :CÔTÉ à la ligne de titre.

Notre procédure sera alors :

```
POUR CARRÉ :CÔTÉ
      RÉPÈTE 4[AV :CÔTÉ DR 90]
FIN
```

A l'appel de la procédure, on donnera une valeur à la variable «CÔTÉ. Essayez avec CARRÉ 40, CARRÉ 10, etc., pour voir comment variera la taille du carré.

Voyons ce qui se passe exactement lorsque vous tapez CARRÉ 30. LOGO lit d'abord la définition de CARRÉ. La ligne de titre lui dit qu'une entrée au clavier est attendue et qu'elle doit s'appeler «CÔTÉ. La valeur sur la ligne de saisie (ici, 30) est allouée à la variable «CÔTÉ, et les commandes de définition de la procédure sont alors exécutées. Lorsque LOGO arrive à la ligne AVANCE :CÔTÉ, il va à la case appelée «CÔTÉ et prend la valeur qu'il y trouve pour la restituer en saisie à AVANCE.

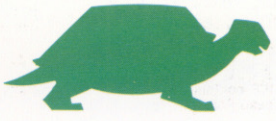
Si une autre procédure utilise la même variable «CÔTÉ, elle devra faire appel à une autre case. «CÔTÉ est appelée en conséquence « variable locale ».

Nous pouvons également avoir recours à la saisie pour des sous-procédures. La procédure MAISON que nous donnons ici est une solution au problème posé dernièrement. Elle peut être modifiée, afin que l'on puisse saisir différentes valeurs :

```
POUR MAISON :GRAND
      CARRÉ :GRAND
      AV :GRAND DR 30
      TRI :GRAND
      GH 30 AR :GRAND
FIN
POUR CARRÉ :TAILLE
      RÉPÈTE 4[AV :TAILLE DR 90]
FIN
POUR TRI :CÔTÉ
      RÉPÈTE 3[AV :CÔTÉ DR 120]
```

CARRÉ est exécuté, AV :TAILLE devenant AV 30. Une procédure similaire est suivie à l'appel de TRI.

Voici une procédure qui trace des polygones, le nombre des côtés étant donné comme entrée :



```
POUR POLY :CÔTÉS
    RÉPÈTE :CÔTÉS [AV 50 DR 360/ :CÔTÉS]
FIN
```

En utilisant cette procédure avec une entrée, POLY 3 tracera un triangle; POLY 4, un carré, etc. Cependant, pour tous les polygones tracés par cette procédure, les côtés auront 50 unités de longueur.

Nous avons utilisé trois noms de variables, «GRAND, «TAILLE et «CÔTÉ. Nous aurions pu prendre le même nom pour les trois, puisque les variables sont locales pour les procédures qui les utilisent.

Pour comprendre comment fonctionnent ces procédures, voyons ce qui se passe lorsque l'on tape MAISON 30. LOGO lit la ligne de saisie et assigne la valeur 30 à la variable «GRAND présente dans MAISON. La première ligne de MAISON est donc maintenant équivalente à CARRÉ 30. La variable «TAILLE dans CARRÉ reçoit à son tour la valeur 30.

Une procédure plus générale pour des polygones supposera deux entrées, une pour le nombre de côtés, une pour la longueur voulue. A cette fin, il suffit d'adapter la procédure POLY pour remplacer 50 par un nom de variable et ajouter ce nom à la ligne de titre :

```
POUR POLY :CÔTÉS :TAILLE
    RÉPÈTE :CÔTÉS [AV :TAILLE DR 360/
    :CÔTÉS]
FIN
```

POLY 5 30 tracera maintenant un pentagone dont les côtés ont 30 pour unité de longueur.

Variables globales

Nous avons vu jusqu'ici des variables limitées aux procédures qui les utilisent. Mais les variables peuvent aussi être définies pour se rapporter à toutes les procédures, ce sont alors des variables « globales ». Elles sont très utiles pour communiquer des informations entre les procédures. Leur inconvénient est de rendre plus difficile le dépistage des erreurs.

La commande FAIRE est utilisée pour assigner des valeurs à des variables globales. FAIRE «CÔTÉ 3 affecte la valeur 3 à la variable «CÔTÉ. FAIRE «CÔTÉ :CÔTÉ + 1 incrémente d'une unité la valeur de «CÔTÉ. La signification exacte de la syntaxe du deuxième exemple est la suivante : trouver la valeur de la variable «CÔTÉ, lui ajouter 1 et remettre le résultat dans la variable «CÔTÉ. Dans les deux cas FAIRE suppose deux entrées, le nom de la variable et la valeur à lui assigner.

Pour résumer les notions de programmation LOGO que nous avons vues dans cet article, nous avons créé des procédures de dessin de spirales. La procédure principale s'appelle EQSPI. Elle suppose trois entrées, la longueur initiale de la ligne à tracer, l'angle à couvrir à chaque tournant de la spirale et un facteur d'échelle affectant la longueur initiale. Différents ensembles de données en entrée seront utilisés pour obtenir plusieurs effets. Nous avons essayé 70 283 0.95, 70 143 0.95 et 20 243 1,05. Essayez d'autres jeux de valeurs pour avoir d'autres effets.

FINENROULE est une nouvelle commande. Elle arrête l'enroulement de l'écran lorsque la tortue atteint les limites de ce dernier. Un message « hors champ » est alors transmis. L'effet d'enroulement donne souvent des résultats graphiques intéressants. Mais pour cette procédure, il gâche l'effet de spirale. C'est pour ce genre de cas qu'existe cette commande, afin de l'inhiber.

La procédure principale EQSPI trace la même ligne de manière répétitive. La longueur de la ligne est déterminée par le facteur d'échelle. EQSPI provoque un tournant selon un angle déterminé et modifie en dernier lieu le facteur d'échelle. La longueur des lignes peut croître ou décroître selon que ce dernier est supérieur ou inférieur à 1. Le nombre assez considérable après RÉPÈTE est destiné à faire s'exécuter la procédure un grand nombre de fois. Si vous voulez l'arrêter, frappez « Contrôle-G » ou « Break ». La plupart des variables sont locales, à l'exception de «ÉCHELLE. Cette dernière est globale du fait que «ACCROISSEMENT change de valeur et que sa nouvelle valeur doit être transmise E.AVANCE. De la sorte, «ÉCHELLE sert à communiquer entre les deux procédures.

Procédure spirale

```
POUR EQSPI :TAILLE :ANGLE :FACT
    MISE
    RÉPÈTE 1000[E.AVANCE :TAILLE DROITE :ANGLE
    ACCROISSEMENT :FACT]
FIN

POUR MISE
    TRACER FINENROULE FAIRE «ÉCHELLE 1
FIN

POUR ACCROISSEMENT :NOMB
FIN

POUR E.AVANCE :DIST
    AVANCE :ÉCHELLE 1 :DIST
FIN
```

Nuances logo

Les versions LCSI utilisent la commande CLÔTURE, plutôt que FINENROULE, pour mettre fin au mode enroulement automatique.
La version Atari utilise FENÊTRE à la place de CLÔTURE. La seule différence est que FENÊTRE, contrairement à CLÔTURE, n'arrête pas la procédure.
Dans la procédure SPIRALE sur Spectrum, remplacer DESSINE, dans la sous-procédure MISE, par CS.

Problèmes sur les procédures

1) Écrivez une procédure pour tracer un cercle de 50 unités de rayon; modifiez-la pour que le rayon soit donné en entrée.
2) Écrivez une procédure qui trace une cible (cinq cercles concentriques).

Réponses aux exercices

1) Les puzzles Tangram

L'homme assis, l'homme qui se penche et le chat, toutes ces formes utilisent l'autre côté du parallélogramme pour le chien de la semaine dernière. Pour retourner un motif avec LOGO, il suffit de changer toutes les commandes DROITE en GAUCHE, et vice versa. Aussi, au lieu de :

```
POUR PAR
     REPETE 2 [AV 25 DR 45 AV 35 DR 135]
FIN
```

nous aurons son image en miroir :

```
POUR PAR
     REPETE 2 [AV 25 GH 45 AV 35 GH 135]
FIN
```

Toutes les autres procédures de dessin de silhouettes sont semblables à celles expliquées ici.

L'homme qui court

```
POUR COURIR
DEPLACEMENT1 TRI1 DEPLACEMENT2 PAR DEPLACEMENT TRI3
DEPLACEMENT4 TRI3 DEPLACEMENT5 CARRE DEPLACEMENT6 TRI1
DEPLACEMENT7 TRI2
DEPLACEMENT
FIN
POUR DEPLACEMENT1
   GH 45
FIN
POUR DEPLACEMENT2
   CL AV 25 DR 135 AV 17.5 GH 45 CB
FIN
POUR DEPLACEMENT3
   CL AV 75 DR 90 CB
FIN
POUR DEPLACEMENT4
   CL DR 90 AV 25 DR 90 CB
FIN
POUR DEPLACEMENT5
   CL AV 50 DR 135 AV 50 GH 135 CB
FIN
POUR DEPLACEMENT6
   CL DR 135 AV 21 DR 135 AV 25 GH 90 AV 50
   GH 90 AV 25 DR 90 CB
FIN
POUR DEPLACEMENT7
   CL AV 25 DR 135 AV 71 DR 45 AR 35 CB
FIN
POUR DEPLACEMENT8
   CL AV 35 GH 90 AV 25 DR 45 AV 17.5 GH 45
   AV 25 DR 135 CB
FIN
```

L'homme assis

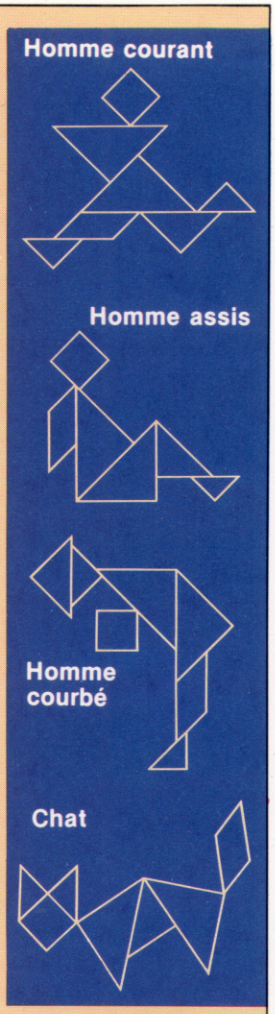
```
POUR ASSIS
   DEPLACEMENT1 TR1 DEPLACEMENT2 TRI2 DEPLACEMENT3 TRI3
   DEPLACEMENT4 TRI1 PAR1 DEPLACEMENT5 CARRE DEPLACEMENT6
   TRI3 DEPLACEMENT7
FIN
POUR DEPLACEMENT1
   GH45
FIN
POUR DEPLACEMENT2
   CL AV 25 GH 45 AV 17.5 DR 90 CB
FIN
POUR DEPLACEMENT3
   CL AR 15 GH 90 CB
FIN
POUR DEPLACMENT4
   CL AV 50 DR 45 AV 25 DR 90 CB
FIN
POUR DEPLACEMENT5
   CL AV 25 GH 45 AV 35 GH 45 CB
FIN
POUR DEPLACEMENT6
   CL AR 50 GH 90 CB
FIN
POUR DEPLACEMENT7
   CL AR 21 DR 135 AR 50 DR 90 AV 35 GH 90
   CB
FIN
```

L'homme courbé

```
POUR COURBE
   DEPLACEMENT1 TRI1 DEPLACEMENT2 PAR1 DEPLACEMENT3 TRI3
   DEPLACEMENT4 TRI3 DEPLACEMENT5 TRI1 DEPLACEMENT6 TRI2
   DEPLACEMENT7 CARRE DEPLACEMENT8
FIN
POUR DEPLACEMENT1
   GH 90
FIN
POUR DEPLACEMENT2
   CL AV 25 DR 135 AV 30 CB
FIN
POUR DEPLACEMENT3
   CL GH 45 AV 35 GH 135 AR 50 CB
FIN
POUR DEPLACEMENT4
   CL DR 90 AV 50 GH 135 CB
FIN
POUR DEPLACEMENT5
   CL DR 90 AV 50 GH 135 AV 5 DR 90 CB
FIN
POUR DEPLACEMENT6
   CL DR 90 AV 25 DR 45 AV 7.5 DR 45 AR 35
   CB
FIN
POUR DEPLACEMENT7
   CL AV 35 DR 135 AV 7.5 GH 45 AV 55 DR 45
   CB
FIN
POUR DEPLACEMENT8
   CL GH 45 AV 36 DR 45 AV 56 GH 135 AV 5
   GH 45 AR 25 CB
FIN
```

Le chat

```
POUR LE.CHAT
   DEPLACEMENT1 TRI3 DEPLACEMENT2 CARRE DEPLACEMENT3 TRI1
   DEPLACEMENT4 TRI1 DEPLACEMENT5 TRI3 DEPLACEMENT6 PAR1
   DEPLACEMENT7 TRI2 DEPLACEMENT8
FIN
POUR DEPLACEMENT1
   CL AV 50 DR 90 CB
FIN
POUR DEPLACEMENT2
   DR 170
FIN
POUR DEPLACEMENT3
   CL DR 90 AV 25 GH 90  CB
FIN
POUR DEPLACEMENT4
   DR 180
FIN
POUR DEPLACEMENT5
   CL DR 90 AV 25 GH 80 AV 50 DR 45 AV 50
   DR 90 CB
FIN
POUR DEPLACEMENT6
   GH 155
FIN
POUR DEPLACEMENT7
   CL GH 160 AV 35 CB
FIN
POUR DEPLACEMENT8
   CL AV 35 GH 45 AV 21 DR 135 CB
FIN
```



Voici les réponses que nous suggérons pour les problèmes de procédures posés dernièrement.

```
POUR MAISON
   CARRE AV 50 DR 30
   TRI1 GH 30 AR 50
FIN
POUR CARRE
   REPETE 4 [AV 50 DR
   90]
FIN
POUR TRI1
   REPETE 3 [AV 50 DR
   120]
FIN
```

```
POUR DAMIER
   REPETE 5 [LIGNE AV
   10] AR 50
FIN
POUR LIGNE
   REPETE 5 [CARRE DR
   90 AV 10 GH 90] GH
   90 AV 50 DR 90
FIN
POUR CARRE
   REPETE 4 [AV 10
   DR 90]
FIN
```

```
POUR SIX.BRANCHES
   TRI DEPLACEMENT TRI EN.
   ARRIERE
FIN
POUR TRI
   DR 30 REPETE 3
   [AV 50 DR 120]
FIN
POUR DEPLACEMENT
   CL AV 29 DR 60 CB
FIN
POUR EN.ARRIERE
   CL GH 60 AR 29
```

Optimisation

Notre cours s'est consacré jusqu'ici à la programmation structurée. Les méthodes décrites permettent des programmes plus faciles à développer, mais n'accélèrent pas l'exécution du BASIC.

Pour que les programmes s'exécutent vite et occupent moins de place mémoire, il est souvent nécessaire de sacrifier la clarté de présentation. Il faut savoir que lorsqu'on optimise un programme, tout ce qui contribue à le faire s'exécuter plus vite le rend inévitablement plus difficile à lire, à comprendre et à mettre au point.

La lenteur inhérente aux langages interprétés, tels que le BASIC, entraîne parfois une lenteur inacceptable d'exécution et contraint à une accélération du programme. La manière la plus efficace de pallier ces défauts est de compiler les programmes BASIC. Mais très peu de micros comportent de vrais compilateurs BASIC. Il existe bien des compilateurs sur disque et cassette sur le marché, mais la plupart concernent exclusivement le BASIC avec nombres entiers et supposent un formatage préalable spécial du programme.

Le compilateur occupe de l'espace mémoire utilisateur et prend d'autant plus de place qu'il est plus efficace et complet. De manière générale, pour les micro-ordinateurs personnels, la compilation n'est conseillée que pour des programmes sans erreurs.

Les accès aux fichiers ralentissent considérablement l'exécution du programme et constituent même la première cause de la lenteur du BASIC. Pour un programme qui effectue fréquemment des lectures/écritures sur disques ou cassettes, les attentes sont inévitables. L'accès à un enregistrement d'un fichier à accès direct sur disquette prend en moyenne un quart de seconde. L'accès aux données de fichiers série prend davantage de temps. Les accès cassette prennent de manière générale plus de temps que les accès disque.

Si ces temps de latence posent des problèmes, essayez de réduire le nombre d'accès et de lire plus de données à la fois. En les sauvegardant en mémoire vive, constituez, également sur fichier, des sauvegardes de mises à jour jusqu'à la fin de la session. Les programmes interactifs posent souvent des problèmes dans la mesure où l'utilisateur doit attendre plusieurs secondes devant son écran. Une solution partielle consiste à réorganiser le programme de sorte que les fichiers soient lus et écrits pendant que l'utilisateur fait autre chose.

Une autre cause de lenteur est l'arithmétique des nombres réels. La présence de décimales fait que la recherche en mémoire d'un nombre réel et son traitement supposent beaucoup plus de cycles machine que pour un nombre entier.

Les programmes qui comportent beaucoup d'arithmétique justifient largement que l'on remplace toutes les variables par des variables entières (SOMME, par exemple, sera remplacé par SOMM%). On peut ainsi gagner jusqu'à vingt pour cent de temps, même sur des programmes relativement numériques. Les applications grandes mangeuses de nombres peuvent, quant à elles, gagner jusqu'à cinquante pour cent de temps.

Écrire un algorithme plus rapide est une des meilleures manières d'accélérer un programme. Nous vous avons déjà donné des références pour vous procurer des algorithmes tout prêts. Essayez-les et soyez à l'affût des nouveaux algorithmes que publient régulièrement les revues spécialisées. La création d'algorithme est une question de créativité et de perspicacité. Le BASIC comporte généralement de nombreuses fonctions incorporées (telles que INSTR, SGN, LOG, etc.).

La vitesse de ces dernières résulte de leur écriture en code machine et du fait qu'elles utilisent les meilleurs algorithmes existants. Les fonctions définies par l'utilisateur, mises en œuvre par la commande DEF FN, s'exécutent également très rapidement. Cette commande est surtout utile pour des programmes avec calculs répétés ou séquences répétées de manipulations de chaînes. La commande DEF FN remplace alors un appel de sous-programme qui est beaucoup plus lent.

Écrire les routines en code machine les rend généralement plus rapides. Les langages interprétés traduisent en code machine les lignes de programme au fur et à mesure qu'ils les rencontrent, et pendant l'exécution du programme.

Vitesse ou présentation ? Tout ce qui contribue à faire s'exécuter un programme rapidement nuit à sa présentation et à sa compréhension. Le bon compromis n'est pas facile à trouver. (Cl. Apple.)

L'écriture en code machine évite cette phase de traduction. Malheureusement, le langage d'assemblage est beaucoup plus difficile que le BASIC. Le coût en temps et en efforts pour apprendre et bien posséder le langage d'assemblage n'est pas compensé par ce que l'utilisateur moyen peut en retirer. Cependant certains programmes, comme ceux qui utilisent du graphisme animé, ne pourraient pas être écrits en BASIC seulement.

Il y a beaucoup d'autres manières de gagner du temps de traitement : utiliser une variable au lieu d'un nombre (par exemple, MAX au lieu de 267,5) pour un accès plus rapide aux valeurs, et spécialement dans les boucles; utiliser différentes lettres pour commencer les noms de variables, et prendre ces lettres à intervalles réguliers dans l'alphabet; utiliser des lignes d'instruction multiples et espacer régulièrement les numéros de lignes (de 10 en 10 par exemple); avec les boucles FOR...NEXT, et si l'interpréteur le permet, se passer de la variable du compteur d'itérations; à l'intérieur d'une boucle, essayer d'éviter de recalculer sans cesse la même valeur, mais la calculer en dehors de la boucle pour l'incorporer en tant que variable.

L'arithmétique des entiers permet de gagner du temps, mais aussi de la place mémoire. Là où il faudrait peut-être 4 ou 5 octets pour sauvegarder un nombre réel, il n'en faut que deux pour un entier. D'autres gains de vitesse représentent aussi des gains de place : l'utilisation de fonctions incorporées ou de fonctions utilisateur supprime du code, tout comme l'écriture en langage d'assemblage et l'utilisation de lignes d'instructions multiples. La compilation a tendance à accroître la taille des petits programmes et ne permet de gagner de l'espace mémoire que pour les grands programmes.

La suppression des REM libère de la place, de même que l'utilisation de chaînes texte plus courtes pour les messages. Sauvegarder hors du programme de gros pavés de texte dans des fichiers les maintient à l'écart lorsqu'ils ne sont pas nécessaires. Retirez autant que cela est permis les espaces des lignes et utilisez des numéros de lignes et des noms de variables plus courts. Si un tableau doit être dimensionné et que sa taille exacte est connue, ne faites pas d'approximation. Attendez de pouvoir disposer de l'information nécessaire pour le dimensionner à l'aide d'une variable, par exemple :

```
10 INPUT«Nombre d'éléments de cette catégorie?»;ÉLÉMENTS%
20 DIM ARRAY%(ÉLÉMENTS)
```

Cela s'appelle « dimensionnement dynamique ». Une autre solution revient à accroître la place réservée au BASIC en mémoire vive. Cela s'obtient par l'intermédiaire de commandes telles que HIMEM. Ces commandes changent la zone mémoire utilisateur réservée aux programmes et aux variables. Le rôle normal de HIMEM consiste à sauvegarder des programmes en code machine de manière à les mettre à l'abri d'une éventuelle réécriture. La même commande peut cependant permettre d'accéder à des positions mémoire supplémentaires en les libérant de leur affectation normale à la mémoire d'écran. Si l'affichage est sans importance, c'est là un bon moyen de gagner 1 K de mémoire. Si l'on ne peut modifier HIMEM, la mémoire d'écran peut néanmoins être utilisée par le biais de PEEK et de POKE s'adressant directement aux positions mémoire de l'écran.

Si toutes ces méthodes ne suffisent pas, on peut encore avoir recours à la commande CHAIN, dont disposent de nombreux BASIC. CHAIN permet à un programme de passer le contrôle à un autre programme. Certains BASIC présentent la commande COMMON qui transmet certaines variables et leurs valeurs courantes au programme suivant. Sur les micro-ordinateurs personnels, CHAIN, lorsqu'elle existe, est généralement une commande très simple, qui permet de transmettre la totalité ou une partie des variables du premier programme au deuxième.

Si les programmes sont écrits selon la programmation structurée, les sous-programmes devraient pouvoir être écrits et testés séparément. Leur exécution peut également être chronométrée séparément. Essayez le simple programme suivant de chronométrage :

```
100 REM Utilisez cette première partie pour
105 REM affecter des valeurs aux variables de
110 REM la routine (n'oubliez pas de
115 REM dimensionner les tableaux et de leur
120 REM attribuer des valeurs virtuelles).
125 REM Ce programme est en BASIC BBC. TEMPS
130 REM est une variable pour le temps
135 REM dont la valeur en centièmes de
140 REM seconde, est donnée par l'horloge du
145 REM système.
200 DÉBUT=TEMPS
210 GOSUB 2000:REM appel de la routine chronométrée.
220 FIN=TEMPS
230 PRINT « L'exécution a pris »;(FIN-DÉBUT)/100;« secondes »
240 END
```

Cette routine permet d'essayer différents algorithmes et diverses façons d'accroître la vitesse.

Comment être rapide

- Pesez longuement le pour et le contre entre un bon style de programmation et le besoin d'un code rapide, même s'il est incompréhensible.
- Si vous le pouvez, compilez; sinon, définissez des fonctions et des procédures.
- Évitez les accès fichier.
- Évitez les nombres réels. Initialisez les variables et utilisez des nombres entiers si votre micro le permet.
- Concevez soigneusement vos algorithmes et étudiez les autres algorithmes.
- Prenez en compte les avantages et les inconvénients du code machine. S'il peut être rapide, il est plus long à écrire et à mettre au point.
- Condensez votre code et retirez vos REM lorsque vous avez obtenu une version de travail.

Changement d'adresse

Examinons maintenant comment les deux registres d'index, X et Y, sont utilisés dans l'adressage indexé. Nous illustrons la valeur de l'indexation en nous référant à plusieurs exemples de programmes.

L'originalité de l'ordinateur à programme stocké résidait dans le fait que, en stockant le programme au même endroit que les données sur lesquelles il doit opérer, celui-ci pouvait se modifier lui-même en cours d'exécution. Son rôle principal n'était pas de modifier les instructions elles-mêmes, mais de changer les adresses où les instructions prennent leurs données. Imaginons que nous ayons accès à un tableau de plusieurs milliers de nombres et que nous devions donner des instructions distinctes pour chacun parce que chaque instruction ne pourrait se référer qu'à une adresse inchangeable.

Ce problème fut grandement facilité par l'introduction du concept de *modification d'adresse*.

De la sorte, la même instruction pouvait être répétée autant de fois qu'on le voulait et renvoyer à des adresses différentes où étaient stockées les données, en utilisant une variable dans un registre pour modifier l'adresse. Nous utilisons tout le temps ce type de concept dans les programmes BASIC.

Par exemple :

```
FOR I=1 TO N
PRINT TABLE(I)
NEXT I
```

Dans ce cas, la même instruction PRINT se réfère à des données différentes chaque fois qu'elle est utilisée, en modifiant l'article de donnée de base (TABLE) et en utilisant une valeur indexée (I), qui est changée chaque fois qu'elle est utilisée.

Le principe fondamental de l'*adressage indexé* est que le contenu du registre d'index est additionné à l'adresse de base donnée dans l'instruction pour fournir l'*adresse effective*. Si cette instruction a lieu à l'intérieur d'une boucle, l'ajustement au registre d'index peut être effectué également dans la boucle.

Le 6809 a non seulement deux registres pour cela, les registres X et Y, mais encore deux autres, S et U. Dans des circonstances spéciales, il est possible d'utiliser aussi le compteur de programme. Pour rendre le sujet de l'adressage indexé encore plus complexe, il y a divers modes d'indexation. Cependant, ceux-ci couvrent presque tous les besoins de la programmation. Nous emploierons dorénavant l'adressage dans tous nos programmes.

L'adressage indexé est indiqué par l'addition de ,X à la zone opérande — si le registre utilisé est X, bien entendu. La forme générale d'une instruction indexée est donc :

```
Opc     Décalé, Registre d'index
LDA     TABLE1,X
STA     TABLE2,Y
```

Dans bien des situations, le décalé est nul, auquel cas il peut être omis. Par exemple :

```
Opc     ,Registre d'index
LDA     ,X
STA     ,Y
```

Supposons que nous ayons un tableau de 64 valeurs à 8 bits stockées en $3000 et que nous voulions accéder aux octets en séquence. Nous pouvons définir l'adresse de base et réserver de l'espace pour le tableau utilisant les directives :

```
        ORG     $3000
TABLE   RMB     64
```

Ces instructions mettent le compteur de programme à $3000, définissent TABLE comme commençant à $3000, et réservent les 64 octets suivants. Nous avons dès lors accès aux octets en utilisant la suite du code : la nouvelle directive ORG signifie que notre code sera stocké dans une autre partie de mémoire que nos données. Quand on commence à utiliser l'adressage indexé, c'est une précaution raisonnable pour éviter à une boucle de sortir du contrôle et de causer un « écrasement » du programme.

```
        ORG     $1000
COUNT   FCB     0
        LDX     #0
LOOP    LDA     TABLE,X
```

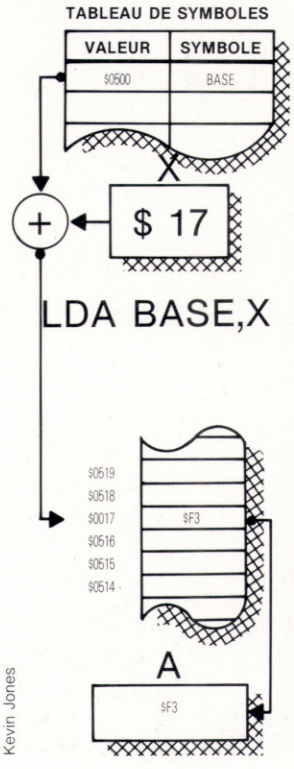
Changeons maintenant la valeur dans le registre X :

```
        TFR     X,D
        ADDD    #1
        TFR     D,X
```

C'est une manière maladroite d'incrémenter X, quoiqu'elle puisse être utile en cas d'incrémentation ou de décrémentation de nombres supérieurs à 2. Le dernier fragment de code incrémente le compte et s'assure qu'il n'est pas égal à 64 (auquel cas le programme se termine et met fin à la boucle) :

```
        INC     COUNT
        LDB     COUNT
        CMPB    #64
        BLT     LOOP
```

Liens d'index
Le symbole BASE — initialisé par $0500 — est l'adresse du premier octet dans un tableau de valeurs. L'instruction de mode d'adressage indexé, LDA BASE,X, prend la valeur de BASE et lui additionne le contenu du registre X pour fournir l'adresse effective de l'octet dont le contenu est chargé dans l'accumulateur. Si cette instruction est à l'intérieur d'une boucle dont X est le compteur, tout le tableau est accessible, octet par octet, en séquence. Puisque X est un registre à 16 bits, la boucle peut parcourir la totalité de l'espace mémoire ($0000–$FFFF dans un système 8 bits comme le 6809).



Kevin Jones

Il y a de nombreuses manières d'améliorer l'efficacité de ce code. L'une des plus utiles est le mode d'*auto-incrémentation*.

L'instruction :

```
        LDA     TABLE,X+
```

aura pour effet d'incrémenter automatiquement la valeur en X après usage. Si nous avons un tableau de valeurs 16 bits, nous écrivons :

```
        LDA     TABLE,X++
```

qui a pour effet d'incrémenter le registre X de 2. Notre boucle de programme initiale est à présent considérablement améliorée :

```
LOOP    LDA     TABLE,X+
        INC     COUNT
        LDB     COUNT
        CMPB    #64
        BLT     LOOP
```

Une autre solution avantageuse consiste à parcourir toutes les valeurs du tableau dans l'ordre inverse, peut-être en utilisant le mode d'autodécrémentation. Cette solution aboutit à une valeur finale nulle dans le registre X, et comme l'autodécrémentation d'un registre d'index met automatiquement les drapeaux dans le registre CC, nous pouvons tester directement la fin de la boucle, sans avoir à utiliser une instruction CPM. Le même effet peut être obtenu en chargeant le registre d'index avec une valeur négative et en l'incrémentant jusqu'à zéro. Chaque fois que l'instruction d'auto-incrémentation est exécutée, elle met les drapeaux de CC pour montrer les résultats de l'incrémentation. Si l'on obtient un zéro, le drapeau de zéro est mis; s'il y a une retenue, c'est le drapeau de retenue, etc.

Il faut cependant rappeler que, en général, il n'est pas toujours suffisant de tester seulement les accumulateurs. Aussi, puisque la plupart des programmes risquent de comporter un traitement entre l'instruction d'incrémentation/décrémentation et le test, il est peu probable que le registre CC demeure inchangé entre l'action et le test.

Si nous décidons de ne pas parcourir le tableau à reculons, on peut néanmoins conserver l'idée du compte à rebours afin de terminer la boucle à zéro. En mode d'autodécrémentation, il ne faut pas oublier que la décrémentation s'effectue *avant* le calcul d'adresse, tandis qu'en auto-incrémentation le registre est incrémenté *après* que l'adresse a été calculée. Ainsi, si X contient 7 et TABLE commence à $1000, l'instruction LDA TABLE,X+ chargera l'accumulateur à partir de l'adresse $1007, puis incrémentera X de 7 à 8. LDA TABLE,-X (notez le signe moins avant le nom de registre), par contre, décrémentera X de 7 à 6, puis chargera l'accumulateur avec la valeur provenant de l'adresse $1006.

En parcourant le tableau à reculons et en tenant le compte dans le registre B, notre boucle devient :

```
        LDX     #64
        LDB     #64
LOOP    LDA     TABLE,-X
        DECB
        BGE     LOOP
```

Le premier de nos deux programmes montre une boucle directe parcourant un tableau de valeurs à 8 bits, dans lequel nous comptons le nombre de valeurs négatives. Le compte dans l'accumulateur B est également utilisé comme décalé d'une valeur fixe dans X. Le second programme montre les deux registres d'index utilisés ensemble, avec un décalé nul. Il fait une copie d'une chaîne de caractères à partir d'un emplacement vers un autre emplacement, où elle sera stockée. La chaîne est de longueur inconnue et se termine par un caractère de retour. Lorsqu'elle est stockée, le caractère de retour est supprimé, et un octet indiquant la longueur de la chaîne est ajouté à son début.

Compteur de valeurs négatives

Un tableau de valeurs à 8 bits est stocké à l'emplacement $3001. Le nombre de valeurs du tableau est stocké en $3000 et est supposé inférieur à 256. Le programme compte le nombre de valeurs négatives dans le tableau.

```
TABLE   EQU     $3000       TABLE contient le nombre de
        ORG     $1000       valeurs.
NEGS    FCB     0           NEGS est là où on garde le
        LDB     TABLE       compte des valeurs négatives.
        LDX     #TABLE      L'adresse du tableau va en X.
LOOP    LDA     B,X         Prend la dernière valeur du
                            tableau.
        BGE     ENDLP       Aller en ENDLP si la valeur
                            est positive.
        INC     NEGS        Sinon, compter si la valeur est
                            négative.
ENDLP   DECB                Compte à rebours sur le
                            tableau.
        BGT     LOOP        Y a-t-il encore des valeurs?
                            Si oui, aller en LOOP.
        SWI                 Sinon, retourner au système
        END                 d'exploitation.
```

Copieur de chaîne de caractères

Ce programme copie une chaîne à partir du tampon d'entrée en $3000 vers le prochain espace de chaîne libre, dont l'adresse est donnée en $0010.

```
CR      EQU     13          Valeur ASCII du caractère de
                            retour chariot.
STRING1 EQU     $3000       Adresse du tampon d'entrée.
STRING2 EQU     $0010       STRING2 contient l'adresse
        ORG     $1000       de l'espace libre.
        LDX     #STRING1    Adresse de chaîne source
                            (c'est-à-dire $3000 d'abord)
                            chargée en X.
        LDY     STRING2     Adresse de chaîne destination
                            (c'est-à-dire adresse stockée
                            en $10,$11) en Y.
        TFR     Y,U         Transférer l'adresse de l'octet
                            de longueur en U.
        LDA     #0          Stocker zéro au premier octet
        STA     ,Y+         de chaîne destination.
LOOP    LDA     ,X+         Prendre le caractère suivant
                            du tampon d'entrée.
        CMPA    #CR         Est-ce un caractère de retour?
        BEQ     FINISH      Arrêter si oui.
        STA     ,Y+         Sinon, le copier (et)
        INC     ,U          Additionner 1 à la longueur.
        BRA     LOOP        Caractère suivant.
FINISH  SWI                 Retour au système
END                         d'exploitation.
```

Dans la course

Vous n'êtes peut-être pas amateur de courses de chevaux. Mais Classic Racing, de Salamander Software, destiné à l'Oric-1 et à l'Atmos, saura pourtant vous retenir.

Un tuyau en or
A tout moment, les joueurs peuvent consulter le programme des courses (image du haut), lequel donne des renseignements précis sur chacune d'elles et signale combien elle peut rapporter. Une fois les chevaux sélectionnés en vue d'une épreuve particulière, il est possible de parier sur tel ou tel animal (deuxième image). Les montures sont alignées en attendant le signal du starter, et les joueurs n'ont plus qu'à contempler la course. La dernière image montre l'arrivée au poteau : le cheval de notre photographe a péniblement terminé quatrième...

Classic Racing fait de vous un entraîneur de chevaux de course le temps d'une saison complète. Le jeu est prévu pour un maximum de six joueurs, mais lorsque ce nombre n'est pas atteint, l'ordinateur prend la place des absents. Vous pouvez par ailleurs fixer vous-même la durée de la saison, qui peut comporter jusqu'à seize réunions différentes, chacune donnant lieu à six courses. Les joueurs peu audacieux pourront toujours choisir un calendrier moins chargé.

L'objectif est simple : gagner le plus d'argent possible. On peut y arriver soit en plaçant un de ses propres chevaux parmi les trois premiers, ce qui rapporte un prix, soit en pariant sur les résultats. Vous devez obligatoirement faire participer une de vos montures à chaque course, mais rien ne vous empêche de parier sur un autre concurrent si vous estimez que cela peut être profitable.

Votre écurie compte seize chevaux, et en début de saison, vous n'avez encore aucune idée précise de leurs mérites respectifs. Toutefois, c'est une époque où il n'y a encore que des compétitions peu importantes; vous pouvez donc vous livrer à des estimations en faisant courir vos bêtes sur des longueurs différentes ou sur des terrains très variables. Il ne s'agit encore que d'essais : voyez comment un cheval se comporte dans des conditions données et établissez votre stratégie en fonction des résultats. Naturellement, cela vous obligera à prendre énormément de notes : à chaque course vous devrez coucher par écrit la distance, le poids du jockey, l'état du terrain et, bien sûr, le résultat final. (Il est regrettable que Salamander Software n'ait pas jugé utile d'intégrer une routine qui aurait géré automatiquement toutes ces données; le jeu en aurait été bien plus simple.)

En vue de la première réunion, vous choisissez les six chevaux que vous y engagerez, et l'ordinateur vous indiquera le nom des concurrents ainsi que le poids qu'ils portent. Ensuite il prendra les paris. En début de saison, les cotes semblent être attribuées un peu au hasard; mais, au fil du temps, elles se font de plus en plus précises.

Les paris peuvent rapporter gros. Un cheval gagnant (ou placé) vaut également un prix en argent; il est donc parfois intéressant de parier sur l'un de vos adversaires, vous pouvez faire ainsi d'une pierre deux coups.

Rien ne vous empêche par ailleurs de monter de véritables « coups », par exemple en engageant un cheval dans une course où, de toute évidence, il n'a aucune chance : une monture qui se comporte bien en terrain lourd et sur une courte distance pourra ainsi être lancée à deux reprises sur un parcours de 2 800 m en terrain sec.

Le but de la manœuvre est de la faire perdre délibérément, donc d'abaisser sa cote, puis de lui faire retrouver son type de réunion préféré. Pourtant, une fois que vous aurez déterminé quelle est la distance et le terrain qui conviennent le mieux à tel ou tel cheval, résistez à la tentation de lui faire courir course après course : il aura besoin de repos si vous voulez qu'il donne toute sa mesure.

Une fois les paris enregistrés, l'action commence pour de bon. Les chevaux prennent la position, le speaker les appelle l'un après l'autre, et la course commence. Le programme donne une bonne approximation sonore des encouragements lancés par les turfistes, tandis que chaque concurrent s'efforce de l'emporter. Rien de plus insupportable que de voir votre candidat s'effondrer à 300 m de l'arrivée, pendant que le favori s'envole littéralement !

Après, les gagnants empochent leurs gains, et le jeu reprend jusqu'à la fin de la réunion. Chacune d'elles propose des distances et des terrains très différents, et si vous estimez qu'un de vos candidats n'est pas à la hauteur, il vous suffira de déclarer forfait lors de trois réunions successives. Vos calculs s'en trouveront simplifiés, mais vous serez contraint de payer une pénalité de 1 000 livres à chaque nouvelle rencontre.

Vers la fin de la saison, la compétition se fait plus difficile. En contrepartie, les récompenses sont de plus en plus importantes : lors du fameux derby d'Epsom, couru lors de la toute dernière réunion, les trois premiers concurrents se partagent une « grosse galette ».

Classic Racing est à ce jour le meilleur programme jamais réalisé pour l'Oric-1 et l'Oric Atmos. Les courses, en particulier du point de vue graphique, sont tout à fait remarquables. Par ailleurs, l'obligation de planifier une saison de courses tout entière donne à ce jeu un intérêt toujours renouvelé.

Il est théoriquement possible de « faire fortune » en une saison.

Classic Racing : pour Oric-1 48 K et Atmos
Éditeurs : Salamander Software
Auteur : Paul Neal
Manche à balai : inutile
Format : cassette

Page manquante
(publicité)

Page manquante
(publicité)